View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by CORE

Wright State University

CORE Scholar

Browse all Theses and Dissertations

Theses and Dissertations

2018

Identification of Nonlinear Constitutive Properties of Damping Coatings

Mackenzie E. Tidball
Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all

 Part of the Mechanical Engineering Commons

Repository Citation

Tidball, Mackenzie E., "Identification of Nonlinear Constitutive Properties of Damping Coatings" (2018). *Browse all Theses and Dissertations*. 2197.
https://corescholar.libraries.wright.edu/etd_all/2197

This Thesis is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.

IDENTIFICATION OF NONLINEAR CONSTITUTIVE PROPERTIES OF DAMPING COATINGS

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Mechanical Engineering

by

MACKENZIE E. TIDBALL
B.S.M.E., Wright State University, 2015

2018
Wright State University

Wright State University
COLLEGE OF GRADUATE STUDIES

December 3, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Mackenzie E. Tidball ENTITLED IDENTIFICATION OF NONLINEAR CONSTITUTIVE PROPERTIES OF DAMPING COATINGS BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Mechanical Engineering.

Joseph C. Slater, Ph.D., P.E.
Thesis Director

Joseph C. Slater, Ph.D., P.E.
Chair, Department of Mechanical and
Materials Engineering

Committee on
Final Examination

Joseph C. Slater, Ph.D., P.E.

Jeffrey Brown, Ph.D.

Harok Bae, Ph.D

Barry Milligan, Ph.D
Interim Dean of the Graduate School

ABSTRACT

Tidball, Mackenzie E. M.S.M.E., Department of Mechanical and Materials Engineering, Wright State University, 2018. *IDENTIFICATION OF NONLINEAR CONSTITUTIVE PROPERTIES OF DAMPING COATINGS.*

Scheduled and unscheduled maintenance actions are a significant cost for gas turbines. Advanced life prediction capabilities help to mitigate these costs. However, the protective thermal and damping coatings applied to the turbine rotors which can help significantly extend component life simultaneously increase the difficulty of modeling the behavior of the components due to the nonlinearity they introduce.

This research addresses modeling and analysis of nonlinear coatings for turbine blades and blisks using a nonlinear constitutive law model for the coating. The theoretical frequency response of the nonlinear system is solved for using the harmonic balance method. The coefficients of the nonlinear constitutive law are then identified by correlating the theoretical nonlinear system responses to experimental data in a minimization function. This approach more accurately identifies the nonlinear coefficients of the system by fitting the model to the full frequency response of the system at resonance as opposed to using just a few points such as with half-power peak picking. The identified nonlinear coating model can later be used in finite element analysis to both optimize the coating within the operating conditions of the turbine and improve life prediction capabilities of the system.

List of Symbols

Chapter 3

σ	Stress
ϵ	Strain
$\dot{\epsilon}$	Strain Velocity
E_1	Coating Modulus
E_2	Strain Squared Nonlinear Coefficient
E_3	Strain Cubed Nonlinear Coefficient
η_1	Coating Damping
η_2	Strain Velocity Squared Nonlinear Coefficient
η_3	Strain Velocity Cubed Nonlinear Coefficient
α_1	High Order Nonlinear Coefficient
M	Moment
y	Distance from the Neutral Axis
A_c	Coating Cross Sectional Area
E_b	Youngs Modulus of the Beam
I	Rotational Moment of Inertia of the Beam
w	Displacement in the y-axis
x	Variable Value Vector
h	Height of Beam
ρ	Density of the Beam
f	Force, N

a_0	Fourier Constant Coefficient
a_n	Fourier Cosine Coefficient
b_n	Fourier Sine Coefficient
ω	Frequency of the Function

Chapter 4

ϵ	Small Positive Value
μ	Damping Constant
ω_0	Natural Frequency
α	Cubic Nonlinear Coefficient
k	Forcing Parameter
Ω	Forcing Frequency
σ	Frequency Detuning Parameter
a	Amplitude of Response Function

Contents

List of Figures

List of Tables

Acknowledgment

I would like to take this opportunity to extend my thanks to my advisor, Dr. Joseph Slater for his guidance and support throughout this research effort.

I would like to thank Dr. Jeffrey Brown for sponsoring this research in conjunction with the Dayton Area Graduate Studies Institute as well as for taking the time to serve on my thesis defense committee.

I would also like to thank Dr. Harok Bae for taking the time to be a committee member for my thesis defense.

Lastly, I would like to thank Tommy George and Bryan Langley for their advice and assistance with equipment and data collection in the TEFF lab.

Dedicated to

Travis and Michelle.

Introduction

This chapter provides a general background on turbine engines and the use of thermally protective coatings for turbine components. Section 1.1 gives an overview of the turbine engine and the conditions experienced by some critical components. Section 1.2 provides information on the protective coatings used to increase the resistance of these components to damage. Section 1.3 outlines some of the attempts to model the coatings in literature.

1.1 Turbine Engines

Gas turbines engines are critical to the American economy. At its basic level, a gas turbine engine converts fuel into mechanical energy. This leads to a variety of applications. For example, gas turbines are essential in power generation. Electricity for homes and businesses is produced by connecting high efficiency turbines to generators. A variety of turbine engines are also used to produce thrust for powering aircraft. The thrust is generated by combusting fuel with oxygen from the atmosphere to create accelerated jet exhaust. Most modern passenger and military aircraft are powered by turbine engines [8]. Even helicopters and some types of tanks use turbine engines.

The function of a turbine engine can be broken down into three processes or sections: the compressor, the combustor, and the turbine.

1. The compressor pressurizes incoming air. The pressurized air is fed into the combustor at extremely high speed.

2. The combustor mixes the high pressure stream of air with fuel and ignites it. The mixture burns at very high temperature. The high pressure, high temperature mixture is then released into the turbine.

3. In the turbine, the mixture expands. This expansion spins the rotating blades, called rotors. This rotation both drives the compressor, and provides mechanical energy for the desired application.

Each of these processes subject the components of the turbine engine to a variety of stresses. The rotors experience large constant stresses from rotational speeds and also from pressures due to flow through the engine. They also experience stresses from vibration induced by passage of stationary components [5]. Because the rotors have so little damping, these vibrations can cause high cycle fatigue, especially near resonance frequencies [17]. The vibration requires mitigation from deliberate mistuning of the rotor blades as well as avoidance of resonance frequencies to prevent blade failure.

Turbine engines are subject to thermal stresses as well. Turbines are Carnot engines; the efficiency and power generated are related to the gas temperature entering the system. The system performance is improved by increasing this gas temperature. Therefore, increases to the system efficiency, effectiveness, and durability all depend on protecting the engine components from an increasingly caustic environment [4].

1.2 Thermal Barrier Coatings Applied to Turbine Blades

Throughout the gas turbine, environmental and operational factors cause damage to the engine. A variety of components are subject to deterioration from the following mechanisms: abrasion, erosion, wear, fretting, and oxidation and corrosion [16]. Due to the large cost of replacing components, mitigation methods based on the application of coatings have been developed.

Thermal-barrier coatings have been developed to protect various metallic engine components from the high temperatures. The combustor, guide vanes, rotors, blade seals, shrouds, and afterburners all benefit from the application of thermal coatings as the gas temperatures are usually higher than that of the nickel alloy component melting point [4].

Areas such as the compressor are subject to erosion by ingested particulates or water droplets. These components are treated with erosion resistant coatings formulated as hard and soft layers; the hard layer will crack upon impact, but the soft layer will stop any crack propagation [16].

Gas turbine blades are also very prone to oxidation and corrosion due to the extreme temperatures they experience as well as the presence of contaminates including sulfates and chlorides. Resistant coatings minimize the effect of oxidation and corrosion by forming a protective oxide coating on the surface [16].

It is well known in literature that in addition to protecting against erosion, oxidation, and corrosion, these resistant coatings can also provide damping to components which are subject to high vibration levels [5]. Specifically, the coatings are a valuable opportunity to add damping to the rotor blades which are otherwise low in damping and subject to vibration fatigue as described in Section 1.1. However, the damping provided by the coatings is also nonlinear with the amplitude of cyclic strain. This makes finite element modeling of the blades for fatigue analysis difficult.

1.3 Finite Element and Fatigue Modeling of the Blades

Due to the high cost of replacing turbine engine components, finite element analysis (FEA) is a critical evaluation tool. This is especially true for rotor blades which are subject to high cycle fatigue. Specifically, FEA is used in conjunction with fatigue metrics so that maintenance actions can be scheduled only as often as necessary but before component failure. Therefore, these models would ideally be as accurate and comprehensive as possible.

For example, the Sines, Crossland, Dang Van and Fatemi-Socie criteria are all evaluated in one effort to develop a fatigue metric for the rotor [14]. However, a damping coating was not included in this analysis. Even when the presence of a coating is otherwise acknowledged, it is not typically included in FEA models, as in this analysis of rotor blade models compared to experimental data [13]. These studies highlight the fact that representing the thermal coating and the associated damping is not standard practice. This is because the model is nonlinear due to the presence of the coating and therefore cannot be solved by the traditional linear methods.

To address this gap, research has been done to develop methods for modeling the coating. One approach is to use a physics based model. For example, one study based their model on point-defect rearrangement which is the dominant damping mechanism for dense oxides [2]. The frequency method and the modal strain energy method were used to estimate damping for a range of temperature values in this study. Unfortunately, this model can only be applied to coatings where point-defect rearrangement is the dominant damping mechanism. Also, the study does not look at variation in damping with cyclic strain. Both of the damping estimation methods employed are linear.

Alternately, FEA models have been used to inform the damping properties of tested materials. One study models a cantilever beam with a coating as a tool for correlating the damping with coating thickness [17]. Another uses a FEA model to back out the Young's modulus and loss factor, again using the modal strain energy method, over a range of test results sampled at different strain levels [19]. Again, these models use linear estimation methods to find the damping, and neither study approaches the problem of reincorporating the damping estimates into FEA models of the blades.

One study was found where the problem of modeling a nonlinear coating with a FEA model has been approached by coding up the harmonic balance method in MATLAB to solve a nonlinear FEA model exported from Ansys [6].

The current research project also proposes to use the harmonic balance method to

solve for the forced response of a coated beam model as well as to identify the nonlinear properties of the damping coatings before incorporating these properties into the model. This research will use existing solvers rather than coding our own in order to attain flexibility so that future users can apply these methods in a way that works best for their system. This approach also leverages the fact that these powerful, comprehensive solvers have already been debugged.

Nonlinear Damping Coatings Literature Review

The following chapter presents the efforts which have been made to characterize the damping of thermally protective coatings. Section 2.1 reviews some of the most prominent linear methods of estimating damping. Section 2.2 provides an overview of the nonlinear damping coating characterizations in literature. The need for further research characterizing nonlinear damping coatings is discussed in section 2.3 and an outline of the current research is presented in section 2.4.

2.1 Damping Characterization Review

In practice there are a variety of ways to describe as well as to calculate damping. The damping ratio, quality factor, and loss factor among others are all parameters used to describe damping. Similarly, there are a variety of ways to calculate damping [3, 9].

The log decrement method is derived from the exponential decay of amplitude in a second order, single degree of freedom, underdamped system. It is assumed that the system is linear. This method is typically employed to estimate the damping of structures subjected to an impulse [3]. This method is subject to error when a variety of damping mechanisms are present in the system which dominate at different amplitudes. For ex-

ample, air damping may be dominant at large deflections and structural damping at small deflections. Therefore, a different answer would be calculated depending on where in the time signal the peaks are measured.

The half-power bandwidth method, on the other hand, takes a damping estimate from the frequency response function (FRF) or transfer function using the peak amplitude frequency and an upper and lower frequency at $\frac{1}{\sqrt{2}}$ of the maximum amplitude. It is an approximate estimate of the bluntness of the resonance peak. Again, the system is assumed to be a low damping, second order, single degree of freedom system. The resonance peak is assumed to be linear and symmetric [3]. This method is inaccurate for highly damped systems. It is prone to overestimating the damping due to poor frequency resolution and leakage from windowing [1].

A FRF fitting method also uses the transfer function to directly fit the data to a second order system model. Like the previous two methods, FRF fitting assumes a linear system. It requires sine sweep tests conducted slow enough to capture the steady state system response [3]. It should be noted that conducting the sine sweep too quickly will yield results which contain the transient response of the system and are therefore less accurate. However, these requirements aren't as much of a concern in high damped systems.

Most of the damping characterizations described in section 2.2 utilize the FRF with the half-power bandwidth method to estimate damping. To capture the nonlinear dependence of the coating on strain, a range of FRFs are taken for a range of strain levels. Then a curve fit is taken of the identified damping values in order to obtain the damping as a function of strain. However, because all of the accepted methods for estimating damping assume that the system is linear, it must be true that the nonlinear coating does not change the shape of the FRF.

2.2 Damping Coating Characterization Research

In order to optimize the damping characteristics of these coating, a wide variety of materials have been evaluated including metals, ceramics, and composite compounds. Samples coated with these materials are evaluated for their stiffness, or storage modulus, and their dissipative capacity, or loss modulus [26]. In a linear system these properties correspond to the real and imaginary components of the Young's modulus. [20] Most current research is directed towards determining the sensitivity of the loss and storage modulus to the coating properties and dimensions as well as external variables such as operating temperature and the amplitude of cyclic strain. For example, a study conducted by Peter Torvik demonstrates the variation of these system properties with strain. The study focused on the various methodologies used to determine the loss and storage modulus, and used samples coated with an air plasma sprayed, titania-alumina blend ceramic coating.

Additional studies demonstrate the dependence of the loss and storage modulus on strain as well as on other variables [19, 21–25, 27, 29]. For example, in an evaluation of titania-alumina ceramic blend coating with an added viscoelastic infiltrate material, it was found that the storage modulus decreases as the coating thickness increases although the loss modulus remains approximately constant. Increased temperature results in a decrease in the storage modulus while changes in the loss modulus with temperature are dominated by the properties of the added infiltrate. [26] A study of the damping abilities of several other thermal barrier coatings for different temperatures found that the damping remained constant for some materials and peaked at distinct temperatures for others. [12] Studies focusing on coating thickness also found varying results. A study of bare titanium plates, plates with a thin coat of magnesium aluminate spinel, and plates with a thick coat of the same found that while the thin coating increased damping in the system, the thick coating did not result in a proportional increase in damping levels. [10]

From this research, it is reasonable to conclude that a comprehensive constitutive law used to model a system with these coatings should be able to accommodate not only the

nonlinear relationship between the loss modulus and the amplitude of cyclic strain, but also variations in temperature and coating thickness.

2.3 Research Requirements

Up to this point, the bulk of the research into nonlinear coatings relies on linear damping estimation methods, such as the half power bandwidth method or modal energy methods [2, 7, 10, 12, 19–21, 25–29]. The damping identification method is typically applied to a sine sweep test on a cantilever beam through resonance. A challenge with this process, and the half power bandwidth method specifically, is quantification of the bandwidth of an asymmetric resonance peak across the strain ranges of interest. Torvik et al. used the Resonant Dwell approach to characterize the loss factor of damped beams at only the resonant peak in order to avoid the challenge of asymmetric bandwidth measurement [25]. This Resonant Dwell approach assumes that the mode shapes are sufficiently linear and that the response is not significantly influenced by other modes or boundary conditions.

This research proposes using a constitutive model of the coating which incorporates nonlinear terms and then fitting the model to the entire experimental FRF. This formulation will be able to model nonlinear phenomena including bifurcations.

This approach was chosen because the initial data collected as shown in section 4.1 is dominated by an apparent spring softening phenomena which causes a ”bent” peak at resonance. This renders traditional estimation techniques inapplicable because they do not account for nonlinear factors as established in section 2.1. With the proposed methodology, any warping of the FRF by the nonlinear coating will only further inform the model.

2.4 Research Overview

To implement this model, a constitutive law allowing for nonlinearity in the stress-strain relationship will be incorporated into the Equation of Motion (EOM) for a cantilever beam. Galerkins Method will be used to discretize the EOM and the harmonic balance method will then be implemented to solve for the forced response.

Experimental data will be collected in the form of sine sweeps on a cantilever beam. Data will need to be collected for both bare beams and beams coated with a thermally protective coating. The bare beam data will be used to identify the Young's Modulus and the damping already present in the uncoated beam using a FRF fitting method. The experimental setup needs to be clear of nonlinear factors which will confound the application of this linear identification technique, so debugging and analysis of the setup is necessary. For this research, debugging of the experimental setup was not achieved in a timely manner. Therefore, it was necessary to simulate the data instead.

The theoretical FRF obtained from the forced response of the nonlinear EOMs will then be compared with the experimental data to extract the coefficients of the constitutive law.

Modeling the Nonlinear System

Modeling the nonlinear damping coating is broken down into two sections. Section 3.1 presents the mathematical model which will be used to derive an EOM for a cantilever beam with an applied coating which can be described using a constitutive law. Traditional methods will not work to find the solution of this nonlinear EOM. Therefore, Section 3.2.1 describes the method which will be used to find the nonlinear solution.

3.1 Coated Beam Equation of Motion

When modeling the nonlinear coating, the first step is to define the constitutive relationship between stress and strain in the system. This is illustrated in equation 3.1 as a power series expansion of Hooke's law. Strain and strain velocity terms up to the third order are included.

$$\sigma = E_1\epsilon + E_2\epsilon^2 + E_3\epsilon^3 + \eta_1\dot{\epsilon} + \eta_2\dot{\epsilon}^2 + \eta_3\dot{\epsilon}^3 + \alpha_1\epsilon^2\dot{\epsilon} + \dots \tag{3.1}$$

The relationship above can be used to model the nonlinear elastic and nonlinear damping properties of the coating. When this law is applied to the experimental strain velocity results, the nonlinear stress-strain relationship can be determined. Note, a more complex plane stress model (with the orthogonal direction) must eventually be used.

In order to identify the unknown terms in the constitutive law, the plane stress model is applied to a cantilever beam in which the governing equation for the system can be found

from the moment produced in that beam during bending. The equation for moment is given by equation 3.2. For the purpose of this analysis, it is assumed that the coating is very thin relative to the beam. Therefore, the integral in equation 3.2 can be taken as equal to the contents of the integral multiplied by the area of the coating layer. Also for this analysis, the y-axis is defined in the vertical direction from the beams neutral axis, and the x-axis is defined along the length of the beam from the clamped end.

$$M = \int_{Area} \sigma y dA \tag{3.2}$$

When solving for the moment, the relationship between strain and beam curvature is also substituted to determine the beam moment in terms of the transverse displacement.

$$\epsilon = y\left(-\frac{d^2w}{dx^2}\right) \tag{3.3}$$

For a beam in which a nonlinear damping coating is applied to each side, the moment is found to be equation 3.4.

$$\begin{aligned} M = {} & E_b I \frac{d^2w}{dx^2} + 2A_c\alpha_1 \left(\frac{h}{2}\right)^4 \left(\frac{d^2w}{dx^2}\right)^2 \frac{d^3w}{dx^2dt} \\ & + 2A_c \sum_{n=1}^{N} \left(\frac{h}{2}\right)^{2n} \left(E_{2n-1}\left(\frac{d^2w}{dx^2}\right)^{2n-1} + \eta_{2n-1}\left(\frac{d^3w}{dx^2dt}\right)^{2n-1}\right) \end{aligned} \tag{3.4}$$

In this equation, the first term is the moment provided by the beam while the remaining terms are the contribution of the nonlinear damping coating. Note that the even terms do not contribute to the moment in the system. This is due to the assumed symmetric nature of the coating on either side of the beam specimen; when the strain and strain velocity squared terms are substituted into the integral of equation 3.2, the coating on the bottom of the beam cancels out the coating on the top of the beam. Therefore, the coefficients E_2 and η_2 fall out for further analysis.

Equation 3.4 can be substituted into the governing equation for the cantilever beam as obtained from Newton's Laws, shown in equation 3.5. [9]

$$\rho A \frac{d^2 w}{dt^2} + \frac{d^2}{dx^2} M(x,t) = f(x,t) \tag{3.5}$$

When combined, these equations form a nonlinear, partial differential equation of motion (EOM) for the system. The Galerkin Method is used to discretize this EOM through the substitution of the first three mode shapes for a cantilever beam. The mode shapes are separated into a time and position component. The now three, expanded EOMs are then multiplied by each of the mode shapes in turn and integrated along the length of the beam. Due to the complexity of these operations, this process is accomplished using symbolic math in Python.

This process results in a set of three, nonlinearly coupled modal equations which are functions of time. They can be solved for the forced response of the first three modes using the harmonic balance method, as described in Section 3.2.1 below. Then, an optimization scheme can be implemented to identify the nonlinear parameters.

3.2 Construction of the Coefficient Identification Code

In order to identify the nonlinear coefficients from the experimental data, the EOMs of the coated beam as described in Section 3.1 are coupled with the harmonic balance method to find the analytical FRF. The harmonic balance method is summarized in subsection 3.2.1, and it's implementation is described in subsection 3.2.2. The experimental and theoretical FRFs are then compared with a bounded minimization function as in subsection 3.2.3 in order to extract the coefficients.

3.2.1 Harmonic Balance Method

In the harmonic balance method, the forced response solution of the system is modeled as a truncated Fourier series in which the coefficients of the terms are unknown. This is shown in equation 3.6.

$$x(t) = a_0 + \sum_{n=1}^{N} a_n \cos(n\omega t) + b_n \sin(n\omega t) \tag{3.6}$$

This solution is substituted into the original EOM and the unknown coefficients are solved for with the users method of choice. To implement this method numerically, the discrete, exponential Fourier series is used as a more functional alternative to the sinusoid-based Fourier series.

The Mousai toolbox in Python implements the harmonic balance method using the discrete Fourier series with the nonlinear solver of choice [18]. The nonlinear solver minimizes the difference between the derivatives provided by a user defined function and the harmonic derivative of an initial guess. The harmonic derivative is calculated using Fourier transforms.

3.2.2 Implementing Mousai

The theoretical FRF generator is essentially a wrapper for inputting the beam parameters into the mousai function in order to rapidly calculate the FRF for a chosen set of nonlinear parameters.

In order to be able to compare exactly with the experimental data, the theoretical FRF generator works across the same range of frequencies and base excitations extracted from the imported data. At each frequency and base excitation value, parameters calculated from these arguments are packaged and sent into the mousai function. When mousai finds a solution, the amplitude and phase output are assigned to a vector for later use.

After a solution is calculated across the frequency and base excitation vectors, the results are post-processed into the total displacements. This process consists of scaling

the contribution of each mode shape based on the measurement location on the beam and summing them. These operations are performed with the solutions in complex form. This process is performed for any one set of chosen nonlinear coefficients. In this fashion, a surface with respect to the frequency and excitation vectors is created rather than just a curve. The surface generated here serves as the theoretical solution for comparison with the experimental data.

While simple, this process is subject to convergence issues due to the numerical issues encountered in the EOMs. Convergence becomes particularly uncertain as the nonlinear coefficients increase. Therefore, this research was unable to simulate highly nonlinear systems. A range had to be defined for each nonlinear coefficient outside of which the convergence issues could not be overcome. Despite this limitation, several approaches were implemented to mitigate convergence difficulties:

First, the second order beam problem was posed as a first order problem in state space form. The reformulation allowed the solver a larger level of flexibility in finding the result. Although the solver runs slower for a more linear system, it converges faster for the nonlinear systems.

Second, an additional function was written to improve the initial guess submitted to mousai. This function is implemented after the FRF curves are solved for. It searches for and interpolates across any gap where convergence failed and then uses the interpolated value as an improved initial guess. Points farthest from the peak are recalculated first. As these solutions are found, the function re-interpolates to further improve subsequent guesses submitted to mousai.

Third, if there are still points where convergence has failed after the recalculation function has run, the interpolated value is recorded as the solution. Although this method introduces more error to the generated FRF, it allows the minimization described in subsection 3.2.3 to proceed.

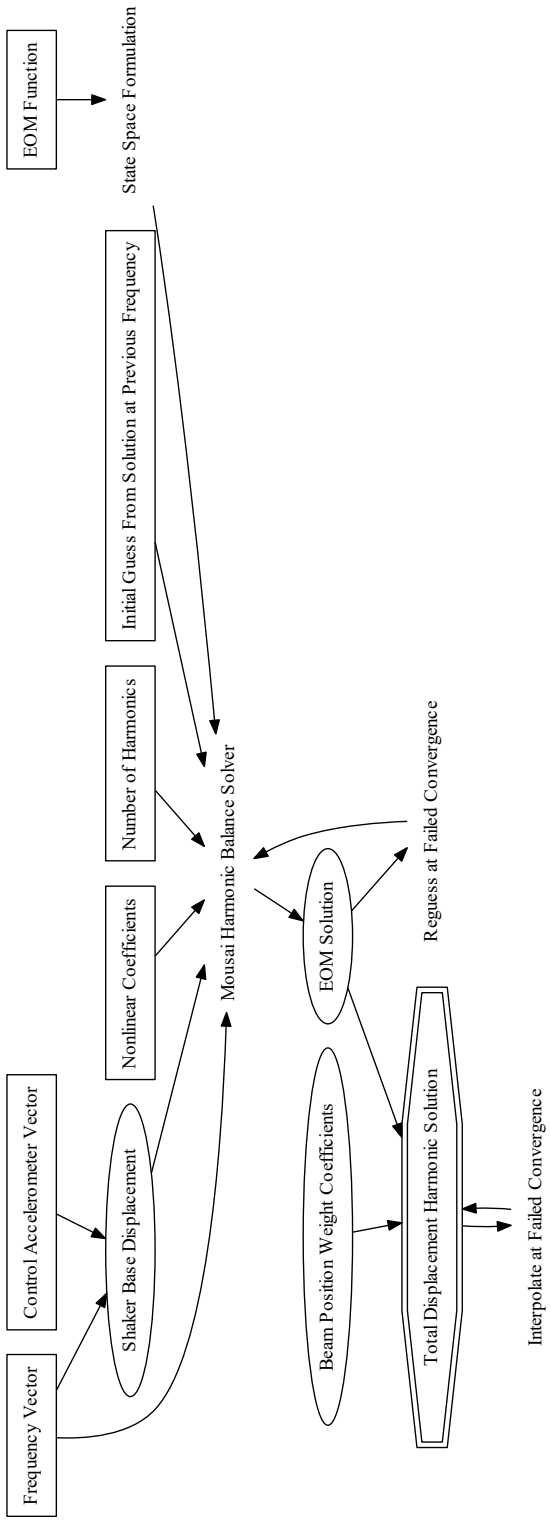

Figure 3.1: Flowchart illustrating the process used to calculate the theoretical FRF of a coated cantilever beam.

3.2.3 Minimization Application to Solve for the Nonlinear Coefficients

With the framework in place to find both the experimental and analytical FRFs, a minimization function can be applied to solve for the nonlinear coefficients. Specifically, the $scipy.optimize.minimize$ python function was used for this research. The method chosen is L-BFGS-B, or a limited memory version of the Broyden-Fletcher-Goldfarb-Shanno algorithm.

The input for this function is the square of the absolute value of the difference between the experimental and theoretical FRFs or $(\Delta y)^2$. At each iteration, the theoretical FRF is recalculated for the new guess of the nonlinear coefficients. The coefficients are adjusted until either the residual hits a minimum value or the gradient of the difference between experimental and theoretical falls below a tolerance. This process is illustrated in Figure 3.2.

In order to prevent the minimization function from evaluating at coefficient values which are too highly nonlinear, bounds are implemented for each. Each coefficient is also normalized to a maximum value of one and a minimum negative value determined by the ability of the system to converge. The minimum bound is no less than negative one. The shape of each nonlinearity is shown in Figures 3.3 through 3.7.

Correlation Between η_3 and α_1 Terms

By examining Figures 3.6 and 3.7, it is apparent that the parameters η_3 and α_1 act on the system in a similar way. Equation 3.1 provides an explanation by showing that they both cubic have a cubic effect. The parameter η_3 is a cubic strain velocity term, and α_1 is the coefficient for strain squared multiplied by the strain velocity. This similarity in the constitutive law results in nearly identical contributions to the beam EOM with the only difference being two strain velocity terms substituted for two strain terms. This can be observed in the system EOM in Appendix C.

In simple harmonic motion, displacement is related to velocity by the frequency where

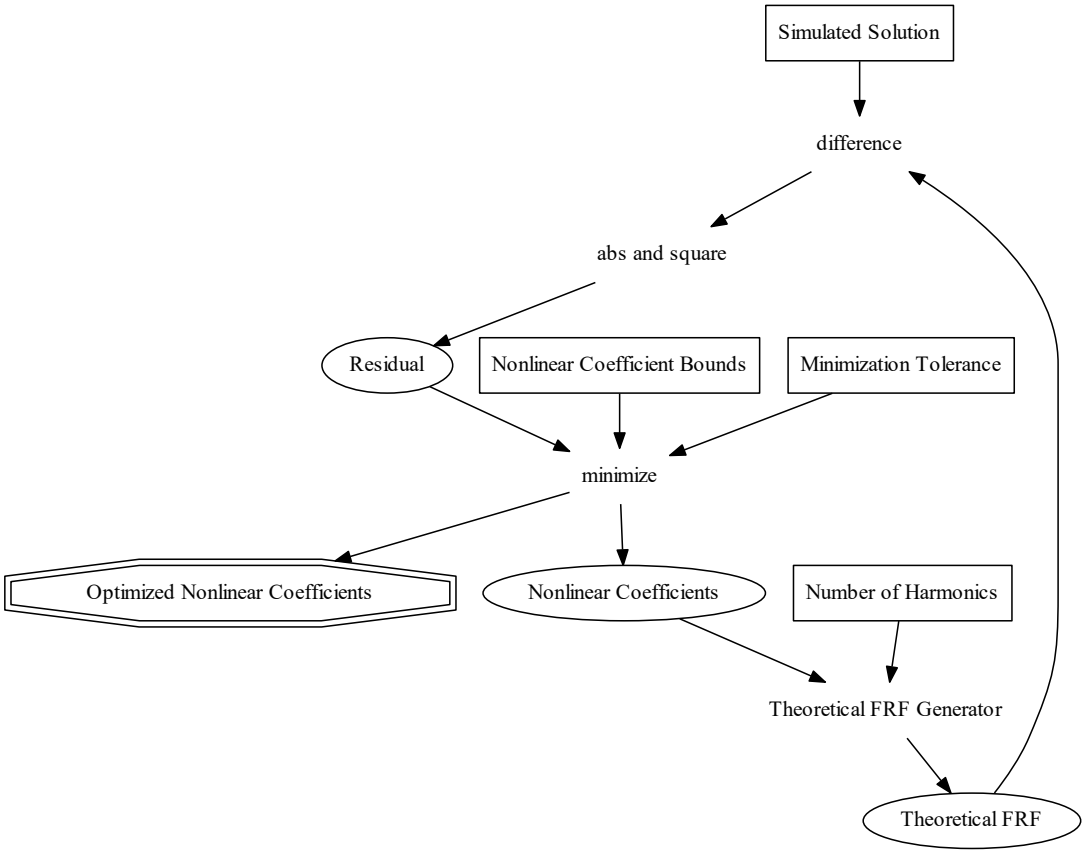

Figure 3.2: Flowchart illustrating the implementation of a minimization function to calculate the optimized nonlinear coefficients of a damping coating

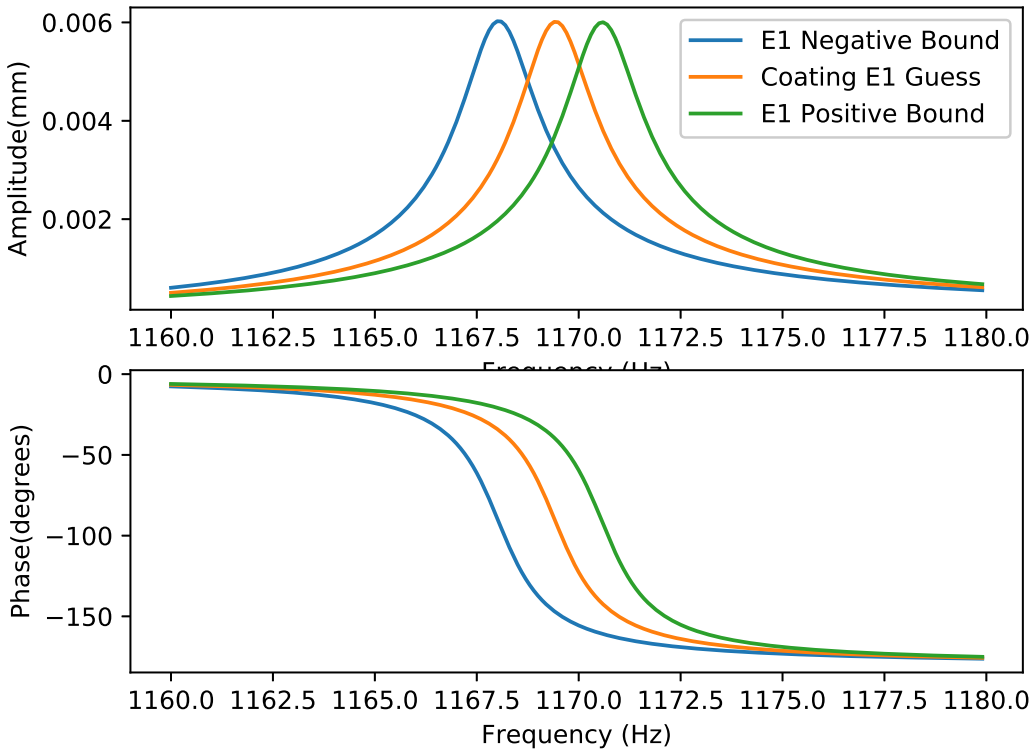

Figure 3.3: Theoretical FRFs within the normalized range of E_1 used in the nonlinear coefficient solver.

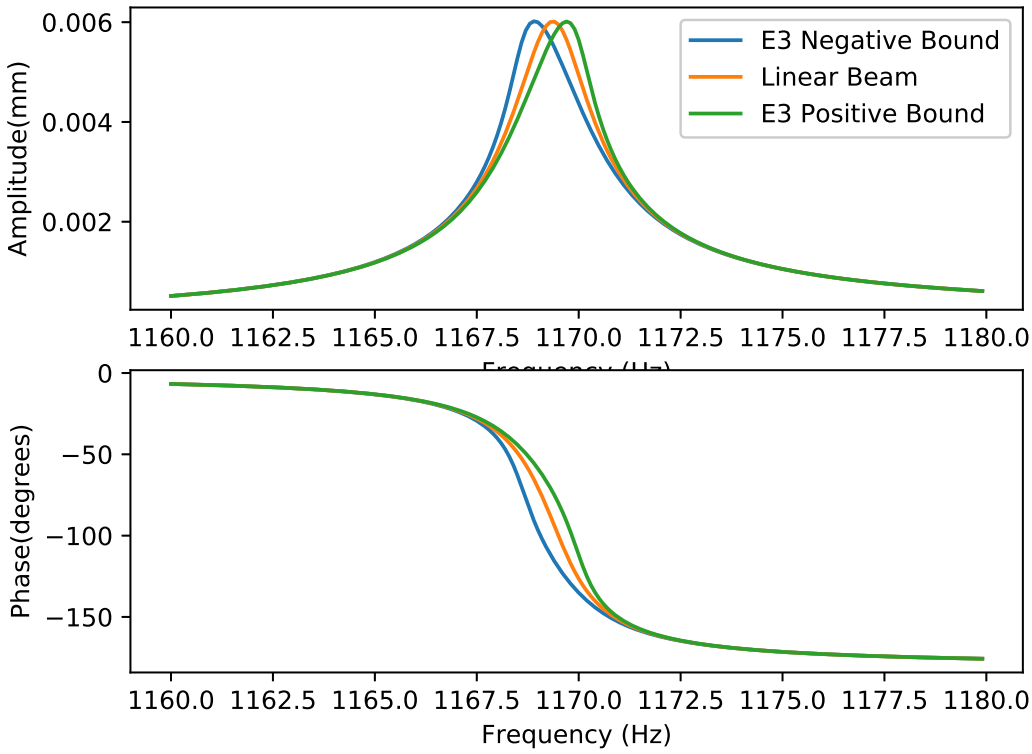

Figure 3.4: Theoretical FRFs within the normalized range of E_3 used in the nonlinear coefficient solver.

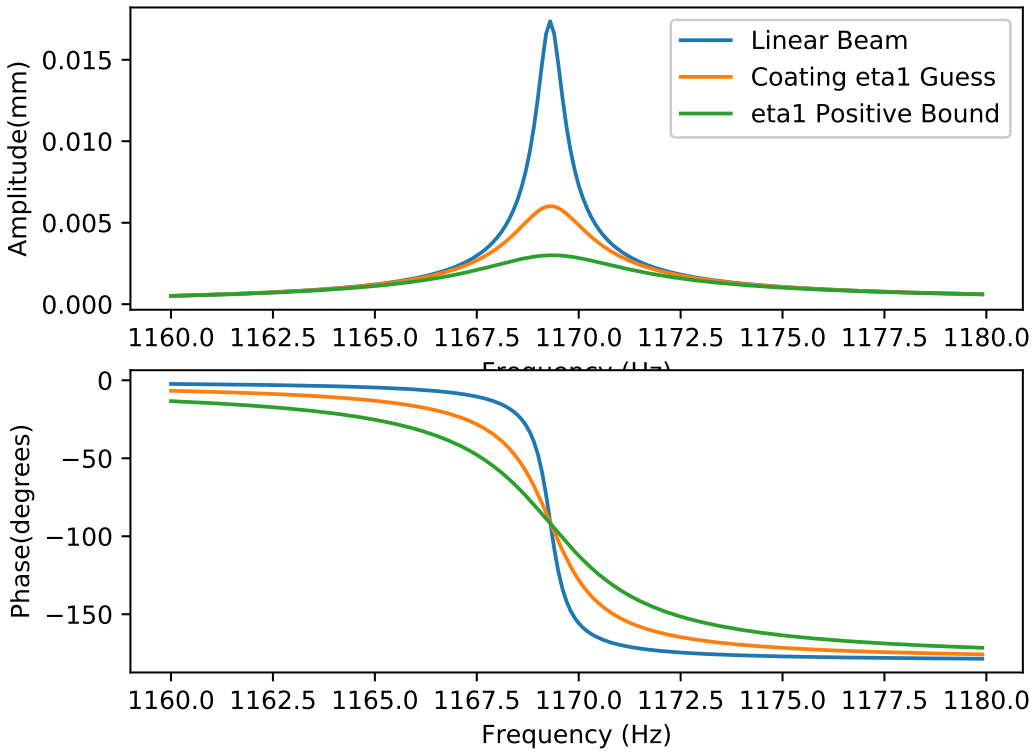

Figure 3.5: Theoretical FRFs within the normalized range of η_1 used in the nonlinear coefficient solver.

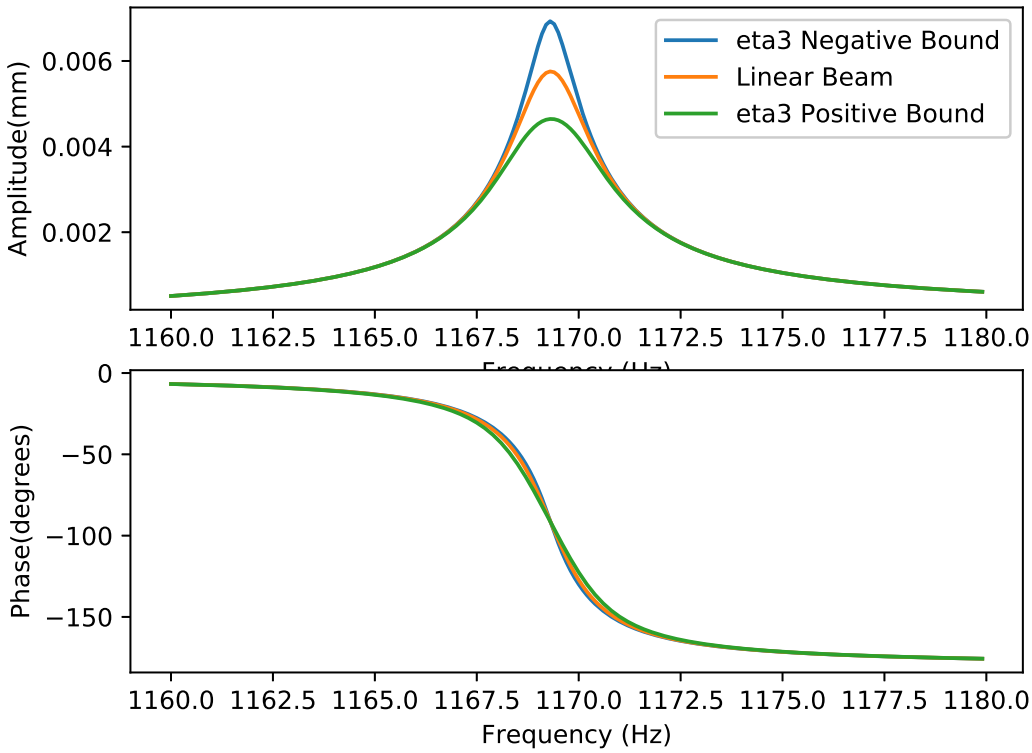

Figure 3.6: Theoretical FRFs within the normalized range of η_3 used in the nonlinear coefficient solver.

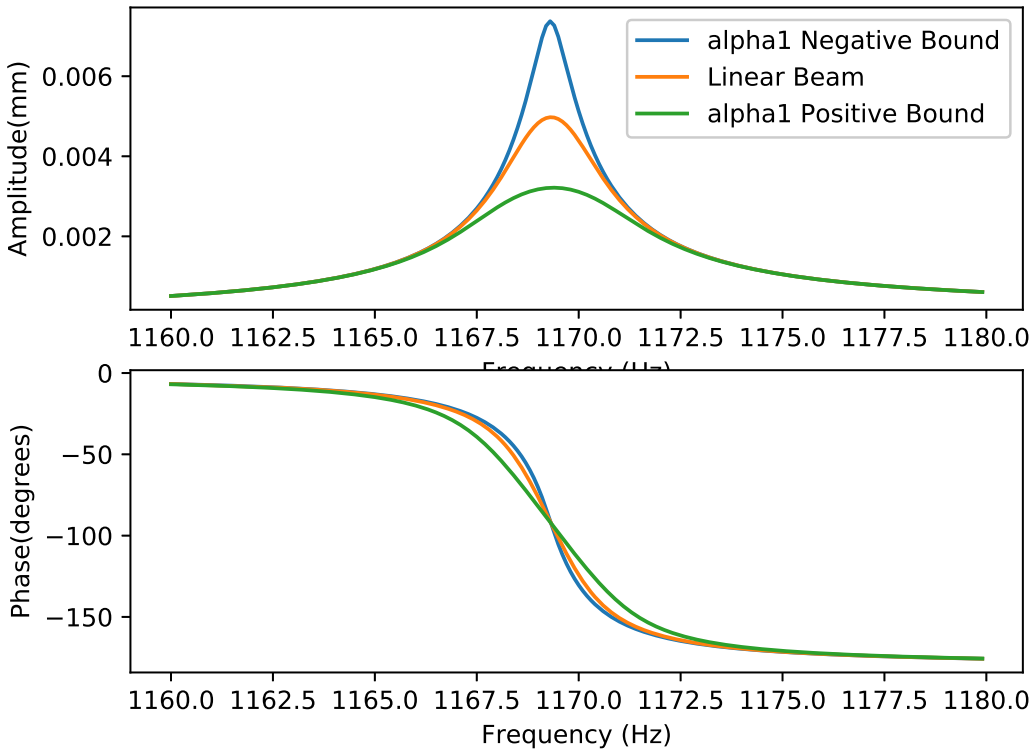

Figure 3.7: Theoretical FRFs within the normalized range of α_1 used in the nonlinear coefficient solver.

$displacement = \frac{velocity}{\omega}$. When this relationship is substituted into the EOM, the η_3 and α_1 terms in the EOM become effectively identical with a frequency multiplier.

$$\alpha_1 = \omega^2(\eta_3) \tag{3.7}$$

Assuming a constant value of η_3, the resulting value of α_1 from this relationship changes as the frequency changes. As α_1 is also assumed to be a constant value, this relationship is only technically true at single frequencies and is thus only useful over a narrow frequency range.

The peak simulated for this research is narrow enough that the minimization function does not distinguish between η_3 and α_1 and finds any arbitrary combination of these variables which collectively contribute a matching amount of cubic damping to the plot. Therefore, only η_3 has been used for simulating data and testing the minimization function with this system. A system with wider peaks would be needed to include α_1 as a distinguishable parameter.

Results

This chapter presents an overview of the approaches taken to this research and the results of the efforts. Work was done in three phases. Section 4.1 describes the analysis initially performed on pre-existing data. Section 4.2 summarizes the efforts which were made to rid the experimental setup of extraneous nonlinear effects. Finally, section 4.3 covers theoretical modeling of the coated beam as well as the extraction of the constitutive law nonlinear coefficients.

4.1 Initial Data Analysis

At the commencement of this project, a significant amount of data from uncoated and coated beams had already been collected in prior research [20, 26]. This made it possible to begin analysis of the data immediately. However, the data for the uncoated beams contained a nonlinear spring softening phenomenon, which was not consequential to prior efforts since their primary purpose was to quantify the magnitude of damping of coated materials. The damping of uncoated beams was very low, with a loss factor on the order of 0.001 for mode 3 at 1500 microstrain. The resonant dwell method used to quantify damping is unaffected by this nonlinear behavior.

The nonlinear behavior of the undamped beam results is shown in Figure 4.1. The presence of this phenomenon renders the standard linear frequency response function curve fit inadequate.

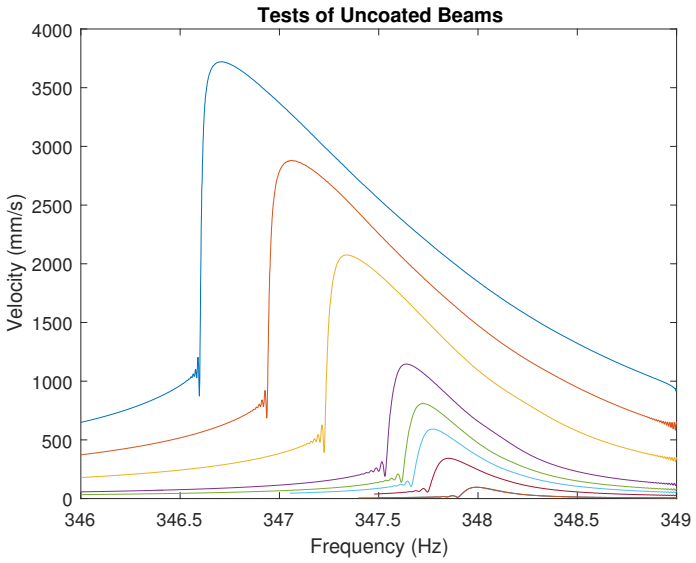

Figure 4.1: Sine sweep results on an uncoated beam through the first bending mode for different excitation levels between 0.1 G and 2.5 G.

In order to obtain a curve fit of the results and quantify the nonlinear parameter, we decided model the beam as a mass normalized, Duffing oscillator softening spring as shown in the equation below. The nonlinear parameter in this model is α. [15]

$$\ddot{x} + 2\epsilon\mu\dot{x} + \omega_o^2 x + \epsilon\alpha x^3 = \epsilon k \cos\Omega t \tag{4.1}$$

Because this model is nonlinear, it is not possible to fit the frequency response data with the augmented Vandermonde matrix obtained from the transfer function in a linear system. In order to obtain the coefficients of the model, an optimization loop was constructed.

This loop splits the resonance peak into two data sets with the demarcation at the maximum amplitude. The correlation between frequency and amplitude are compared by substituting the amplitude values into the equation below and comparing the output to a sigma vector obtained from the associated frequency vector. Sigma is a detuning parameter which relates to the frequency vector by providing variation about the natural frequency.

The amplitude values for the lower frequency data set are used with the version of

the equation below where the radical is subtracted. The amplitude values for the higher frequency data set are used with the version of the equation where the radical is added.

$$\sigma = \frac{3}{8}\frac{\alpha}{\omega_o}a^2 \pm \frac{1}{2}\sqrt{\frac{k^2}{\omega_o^2 a^2} - 4\mu^2} \tag{4.2}$$

$$\Omega = \omega_o + \epsilon\sigma \tag{4.3}$$

The optimization loop minimizes the difference between the two sigma vectors obtained using the amplitude vector and the frequency vector respectively by finding appropriate values of the damping, natural frequency, nonlinear parameter, and a mass scaling value. The loop performs this for both data sets simultaneously.

The resultant curve fit for a sample data set is shown in Figure 4.2.

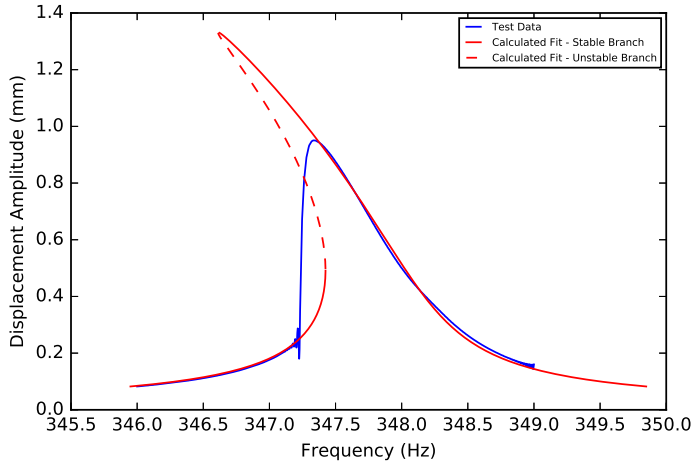

Figure 4.2: Curve fit of the uncoated beam sine sweep results using a nonlinear Duffing oscillator, spring softening model. The experimental curve was obtained by sweeping through the second mode at a base amplitude of 2.5 G.

After working on this model, several features of the test data inconsistent with the mathematical model became apparent. These include an early 'jump' from the top solution to the bottom solution. Such an occurrence would not be so surprising were it not for an

apparent downturn at the tip of the peak. Such a soft peak would most likely indicate that the sweep was reaching the end of the top solution. It would be unexpected to find it in a case where the beam switched between solutions early.

An additional concern is the oscillation observable at the base of the left hand side of the peak. This oscillation would most likely indicate that the sine sweep was not conducted slow enough for the system to fully react.

4.2 Experimental Testing

To explore the inconsistencies between the Duffing oscillator model and experimental data, a variety of tests were conducted on the original test setup with the goal of obtaining a linear FRF. The original goals of the additional testing also included establishing a slow enough sweep rate such that the oscillation was not present in the sweep. It was also desirable to determine the change in the nonlinear behavior of the system at lower amplitudes. The governing theory was that the nonlinearity in the results was a geometric nonlinearity. Therefore, the ultimate goal was to determine an amplitude of excitation low enough that the response of the beam was effectively linear. The parameters of the linear system can be obtained from this response and could then be included in the optimization loop. This would help find a more accurate estimate of the nonlinear parameter.

4.2.1 Identifying the Source of Nonlinearity in the Experimental Setup

After investigation, it was found that the sweep results are nonlinear even at incredibly low excitation amplitudes. Additionally, a sweep up through resonance yielded a peak which did not contain the 'jump' phenomenon of a Duffing Oscillator model and which did not align as expected with the sweep down. This can be observed in Figure 4.3.

These observations made it a question as to whether or not the nonlinearity is a geo-

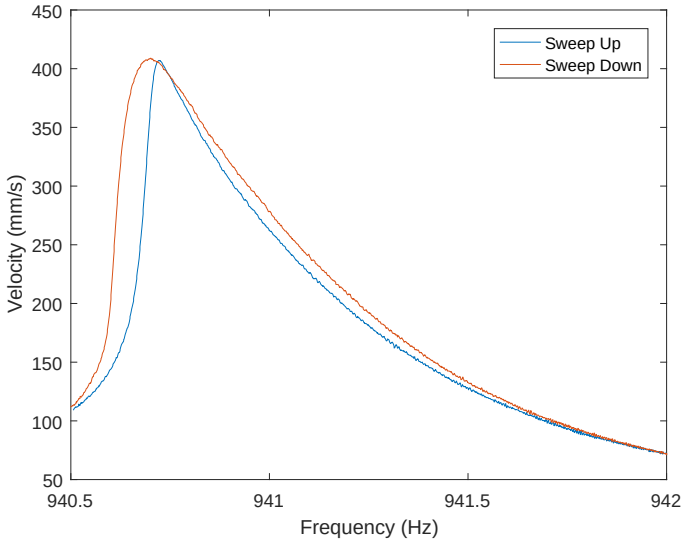

Figure 4.3: Plot of an uncoated beam sine sweep down and then up through the resonance

metric nonlinearity in the beam or if it is a result of some aspect of the test setup. In order to explore this uncertainty, a variety of checks were performed on the system.

System Checks on the 6,000 lb Shaker

System checks on the six thousand pound shaker proceeded as follows:

Additional accelerometers were added to the clamp and the shaker head to ensure agreement between the signals. Although there was some disagreement between the shaker head and the clamp, it was determined to be insignificant.

It was additionally hypothesized that the magnetic field generated by the shaker could be affecting the accelerometer signal. However, a non-contact laser focused on the top of the accelerometer verified it's results.

It is known that large shakers can have an issue with wobble from an unbalanced shaker head. A secondary block the same size as the clamp had been mounted on the opposite side of the shaker head throughout the beam testing performed at RQTI in order to avoid this. As a verification, the movement of the shaker head was measured by tracking

multiple points on the surface. The results showed that the vertical oscillation of shaker head is almost entirely in phase.

Throughout these checks, nothing was found to be inconsistent with expectations in the operation of a shaker. Finally, an impact test was conducted to isolate the reaction of the beam from any interaction with the test setup. While still mounted in the clamp on the shaker, the beam was impacted with a hammer in order to introduce a significant amount of energy into the system. Geometric nonlinearity should be apparent for such a large excitation if it is a factor. The results are shown in Figures 4.4 and 4.5.

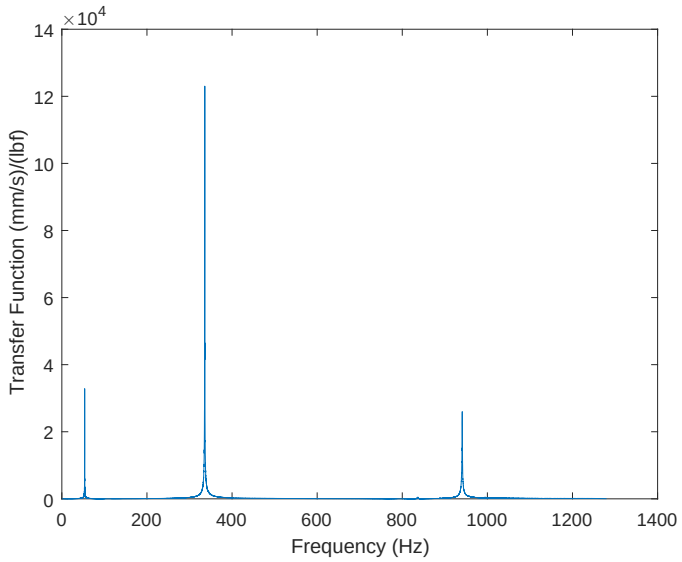

Figure 4.4: Results of a ping test performed on a bare beam specimens

These results demonstrate a linear response. The curve appears to be symmetric and is approximated well with a linear curve fit. Additionally, the coherence is approximately unity throughout the frequencies at resonance. It is expected for a nonlinear beam that the coherence would drop due to the multiple solutions possible at a single frequency. Lastly, only the three beam bending modes are apparent. Sub-harmonics and super-harmonics would be present at integer multiples of the natural frequencies for a nonlinear system.

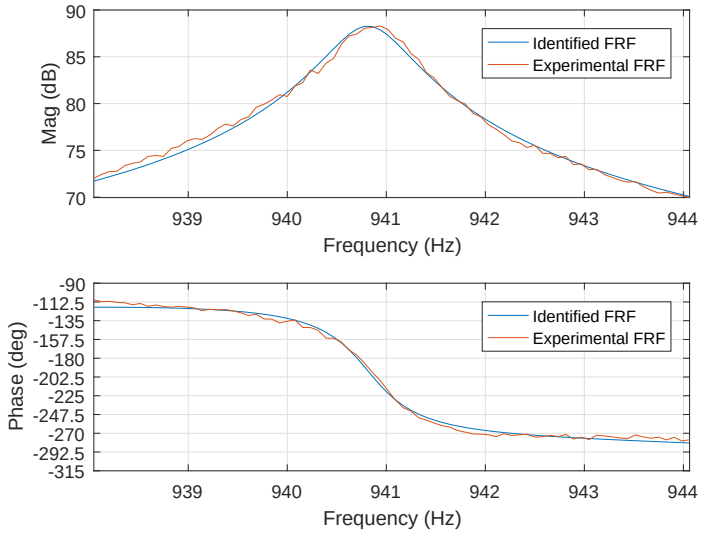

Figure 4.5: Results of a curve fit on the third bending mode of a ping test.

Testing Using an Alternate Shaker

In order to determine whether the nonlinearity follows the beam or the shaker, the beam and clamp were moved to an alternate setup for additional testing. The clamp and beam were mounted on large plate which was driven by a small piezoelectric shaker also mounted to the plate surface. The setup was tested with the plate bolted to an optical table.

The system is nonlinear in this bolted configuration. The response is almost identical to what was observed on the six thousand pound shaker. It was also theorized that inserting a rubber pad between the plate and the table could eliminate possible nonlinearities at the interface. However, mounting the plate on a thin rubber pad demonstrates the same behavior as the metal-to-metal configuration.

Testing on the alternate setup yields several conclusions. The first is that the nonlinearity appears to follow the beam and clamp rather than the shaker. This corroborates the evidence found in testing on the six thousand pound shaker that neither the electromagnetic field generated nor any possible wobble from the large shaker is the culprit. The nonlinearity is also not specific to any one beam.

Testing with a Wright State University Aluminum Clamp

With this information, the final variable to check was the clamp. Testing was moved to a setup at Wright State University (WSU) which had yielded linear results in the past.

Initial testing at WSU was promising. The sine sweep curves appeared to be linearized. However, the only sensors available at WSU are traditionally mounted accelerometers. It was determined that they could be adding a significant amount of damping to the bare beam system as compared to the laser vibrometer sensor. Therefore, the clamp used at WSU was moved back to the lab at RQTI for verification of results.

First, only the laser vibrometer was switched out for the accelerometer in the RQTI setup. This switch did show that the traditionally mounted accelerometer does linearize the results by adding damping. Therefore, the results obtained at WSU are not suitable for analysis.

Second, the steel clamp used by RQTI was switched for the aluminum clamp at WSU with the laser vibrometer as a sensor. It was theorized that the softer material of the aluminum clamp would deform slightly when clamped on the beam and provide a better clamped interface.

As is shown in Figure 4.6, the aluminum clamp does in fact significantly linearize the beam behavior. When the beam was tested on the steel clamp, the spring softening phenomenon was present even on the lowest excitation sine sweep tests. On the aluminum clamp, the same sine sweep test at 0.1 G which was highly nonlinear on the steel clamp appears linear on the aluminum clamp.

4.2.2 Collection of Experimental Data

This improvement was considered significant enough to proceed with data collection. Data for seven coated beams already tested by RQTI has been collected for an excitation frequency range of 0.05 G to 2.5 G. Data for one of these beams is plotted in Figure 4.7.

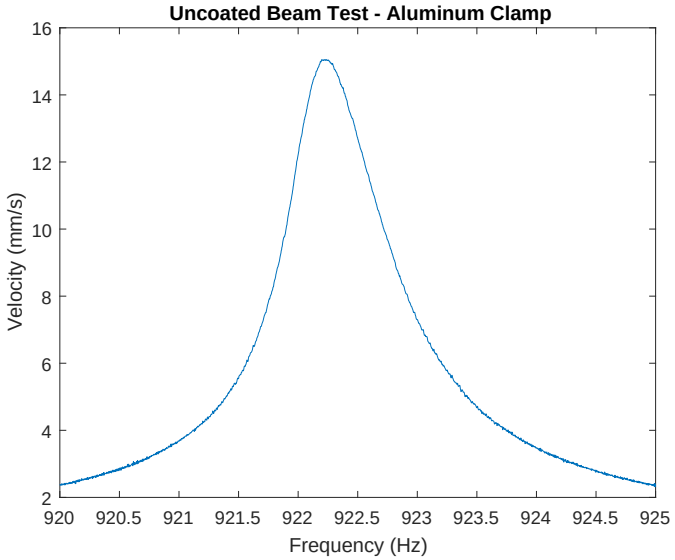

Figure 4.6: Sine sweep results on an uncoated beam through the second mode at an excitation level of 0.1 G when tested on an aluminum clamp.

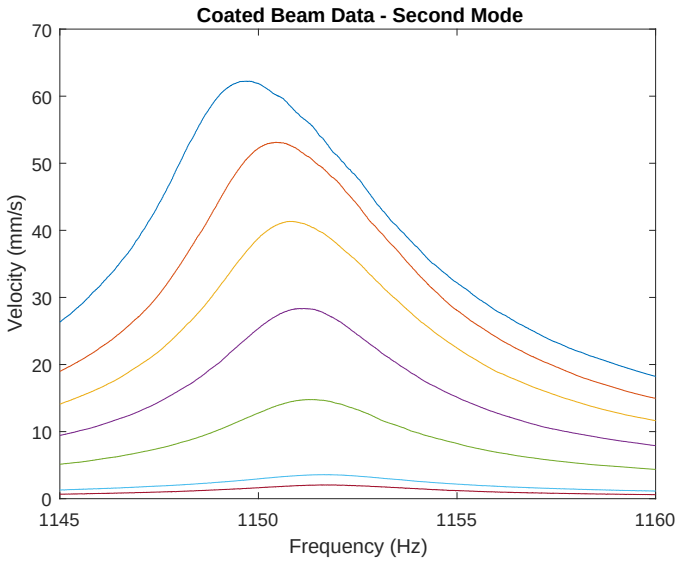

Figure 4.7: Sine sweep results on a coated beam through the second mode for different excitation levels between 0.05 G and 2.5 G when tested on an aluminum clamp.

From Figure 4.7, it can be observed that a nonlinear spring softening effect and a frequency drift is observed for sine sweep curves above 15 mm/s. Presumably, this effect is caused by the coating. However, looking at Figure 4.6, the sine sweep conducted on the uncoated beam does not rise above 15 mm/s. Therefore, we cannot be absolutely certain that the nonlinear effect is not still manifesting due to the boundary condition without conducting more testing.

Because of this uncertainty, it was decided to simulate the sine sweep data for testing the proposed coefficient identification methodology. Simulating the data offers two main benefits. First, this approach eliminates the possibility of any nonlinearity in the setup which does not stem from the coating. Second, this approach allows the analysis of a full test matrix of data. Data can be simulated for a range of coating parameters and for a range of noise in the data. Then, the reliability of the coefficient identification code can be investigated.

4.3 Simulating Data

After the decision was made to simulate the sine sweep curves, efforts in this research project turned toward building a computer model of the coated beam. First, the three EOMs for the first three mode shapes were solved for using the methodology described in Section 3.1 and a symbolic python toolbox. These EOMs were built into a function which was then used to simulate the sine sweep curves needed to test the methodology. The results of this are used as the experimental FRFs.

The sine sweep curves were generated using a methodology based on actual, standard sine sweep tests. Rather than using the shaker, a time signal is generated using the $scipy.integrate.odeint$ function in python as well as the function containing the nonlinear beam EOMs. Starting at an initial guess of zero, the integration proceeds until the system reaches a steady state. Noise is added to the resultant time signal using a random number

generator to introduce error in the results thereby simulating actual noise in the sensing device. This added noise will be included in a test matrix for robustness testing of the coefficient identification code.

Finally, a fft is taken of the last cycle to find the primary harmonic of the signal. The magnitude and phase are the absolute value and the angle of that complex number respectively. It is important to note that in order to correctly calculate the phase, the time vector used in the integration function is calculated to be consistent with an even number of cycles for any given frequency.

The results are then post processed in the same fashion as described in subsection 3.2.2. This resulting surface has been used as the experimental data for the purposes of this project. It should be noted that the beam dimensions and parameters such as the Young's modulus and damping constant were based on the already existing test samples used in Sections 4.2.

4.4 Results of Minimization Application

For this research effort, the effectiveness of the coefficient identification code was tested for reliabiliy against the variation of several different parameters. Two of these parameters were obviously the nonlinear coefficients: $E3$ and η_3. Five values of E_3 between a normalized value of positive and negative one were used to account for both spring hardening and spring stiffening. Three values of η_3 between a normalized value of zero and one were used. Negative values for η_3 were disregarded as these would indicate negative cubic damping and an unstable system.

Although results for E_1 and η_1 were also calculated and are shown, they were not included in the test matrix because these are the linear stiffness and damping components of the coating. It is assumed a reasonable guess for these will be made prior to using the identification code. For this analysis it was decided that $E_1 = 0$ and $\eta_1 = 0.39$.

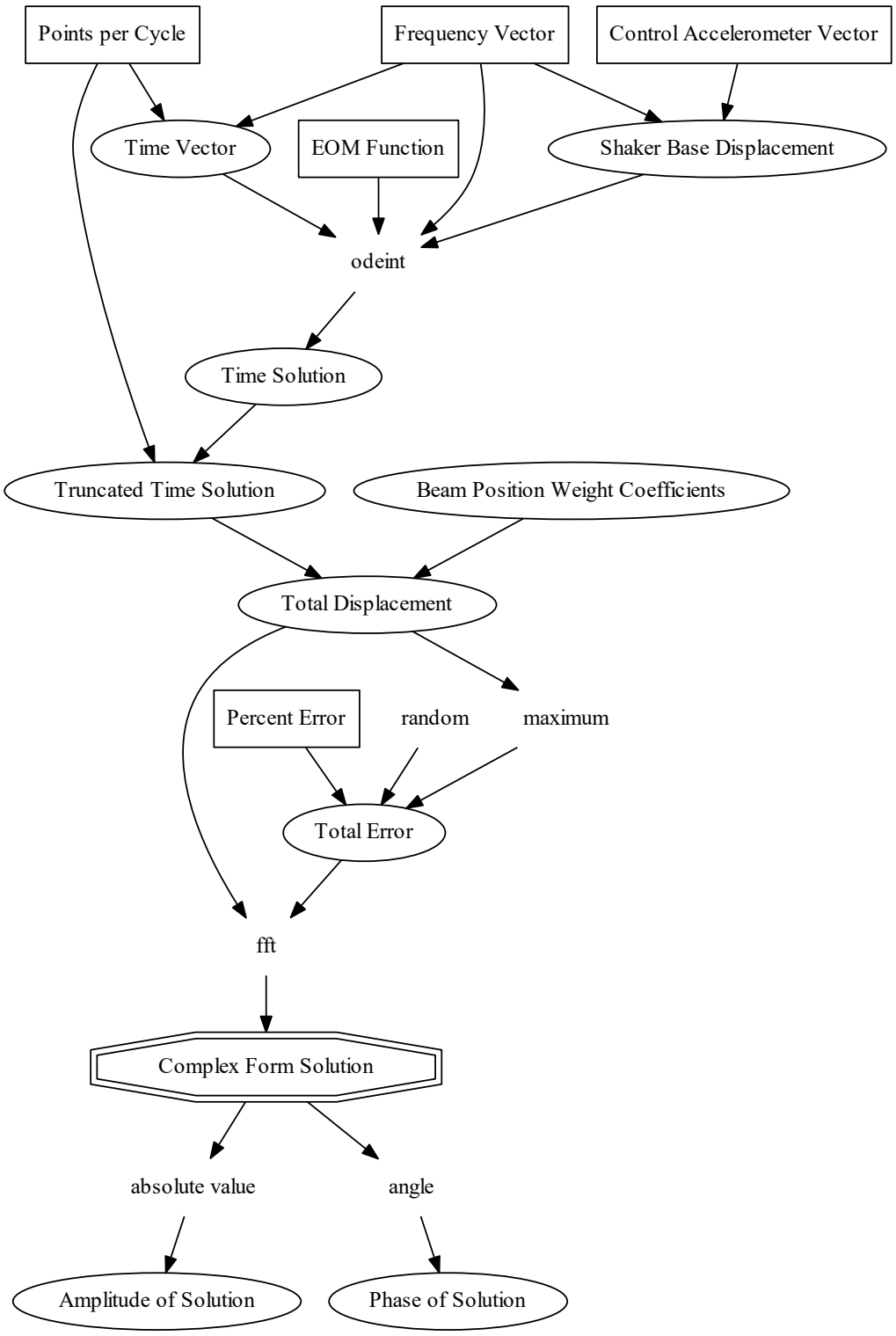

Figure 4.8: Flowchart illustrating the process used to simulate sine sweep data of a coated beam.

The final parameter which was examined is assumed noise in the accelerometer. Data was simulated for percent noise between 0% and 10% with 2% increments.

The results for this test matrix are shown in table 4.4 through table 4.4. The difference values plotted are the difference between the identified normalized parameter value and the actual normalized parameter value. They are organized into cases which are sorted first based on η_3 and then based on E_3 as shown below.

Case Number:

1. $E_3 = -0.9, \eta_3 = 0$
2. $E_3 = -0.5, \eta_3 = 0$
3. $E_3 = 0, \eta_3 = 0$
4. $E_3 = 0.5, \eta_3 = 0$
5. $E_3 = 0.9, \eta_3 = 0$
6. $E_3 = -0.9, \eta_3 = 0.5$
7. $E_3 = -0.5, \eta_3 = 0.5$
8. $E_3 = 0, \eta_3 = 0.5$
9. $E_3 = 0.5, \eta_3 = 0.5$
10. $E_3 = 0.9, \eta_3 = 0.5$
11. $E_3 = -0.9, \eta_3 = 0.9$
12. $E_3 = -0.5, \eta_3 = 0.9$
13. $E_3 = 0, \eta_3 = 0.9$
14. $E_3 = 0.5, \eta_3 = 0.9$

Table 4.1: Identified nonlinear coefficients for 0% error.

Case Number	E_1	E_3	η_1	η_3
1	0.0078	-0.9509	0.3858	0.0702
2	0.0008	-0.5015	0.3966	-0.0052
3	0.0006	0.0000	0.3954	0.0002
4	0.0005	0.5014	0.3967	-0.0062
5				
6	0.0010	-0.8993	0.3969	0.4877
7	-0.0027	-0.4741	0.3943	0.5115
8	0.0016	-0.0077	0.3965	0.4948
9	0.0003	0.5022	0.3974	0.4877
10	0.0135	0.8108	0.3803	0.6312
11	0.0055	-0.9327	0.3966	0.9000
12	0.0060	-0.5464	0.3989	0.8736
13	0.0015	-0.0087	0.3965	0.8970
14	-0.0035	0.5156	0.4037	0.8122
15	0.0085	0.8312	0.3941	0.9000

Table 4.2: Identified nonlinear coefficients for 2% error.

Case Number	E_1	E_3	η_1	η_3
1	-0.0013	-0.8921	0.3973	-0.0121
2	0.0018	-0.5044	0.3953	-0.0007
3	0.0010	0.0002	0.3963	0.0019
4	-0.0040	0.5372	0.3835	0.0680
5				
6	-0.0095	-0.8202	0.3966	0.4872
7	-0.0043	-0.4710	0.3951	0.5044
8	0.0015	-0.0034	0.3988	0.4762
9	0.0580	0.0826	0.4515	0.0891
10	0.0056	0.8737	0.4009	0.4668
11	-0.0025	-0.8757	0.3948	0.9000
12	-0.0013	-0.4902	0.3965	0.8965
13	0.0076	-0.0656	0.4017	0.8558
14	-0.0052	0.5351	0.4007	0.8542
15	0.0188	0.7500	0.4007	0.8402

Table 4.3: Identified nonlinear coefficients for 4% error.

Case Number	E_1	E_3	η_1	η_3
1	-0.0033	-0.8914	0.3879	0.0357
2	0.0020	-0.5016	0.3932	0.0076
3	0.0014	0.0004	0.3972	0.0030
4	-0.0042	0.5262	0.3856	0.0533
5				
6	0.0024	-0.9048	0.3979	0.4760
7	-0.0065	-0.4609	0.3957	0.4983
8	0.0019	-0.0025	0.4007	0.4610
9	-0.0087	0.5408	0.3934	0.5348
10	0.0139	0.7995	0.3909	0.5842
11	-0.0023	-0.8739	0.3970	0.8587
12	0.0050	-0.5492	0.4005	0.8600
13	0.0082	-0.0777	0.4062	0.8242
14	-0.0043	0.5272	0.4012	0.8510
15	0.0043	0.8876	0.3947	0.9000

Table 4.4: Identified nonlinear coefficients for 6% error.

Case Number	E_1	E_3	η_1	η_3
1	-0.0144	-0.8268	0.3975	-0.0202
2	0.0043	-0.5110	0.3922	0.0102
3	0.0018	0.0009	0.3982	0.0037
4	-0.0049	0.5219	0.3969	-0.0138
5				
6	-0.0015	-0.8756	0.3814	0.6389
7	-0.0089	-0.4519	0.3961	0.4948
8	0.0001	0.0132	0.4029	0.4444
9	-0.0120	0.5442	0.3990	0.4954
10	0.0047	0.9000	0.3724	0.7080
11	-0.0033	-0.8607	0.4004	0.8141
12	0.0002	-0.5209	0.3992	0.8711
13	0.0101	-0.1011	0.4118	0.7842
14	-0.0048	0.5238	0.4022	0.8426
15	0.0082	0.8664	0.3916	0.9000

Table 4.5: Identified nonlinear coefficients for 8% error.

Case Number	E_1	E_3	η_1	η_3
1	-0.0071	-0.8617	0.3838	0.0502
2	0.0039	-0.5125	0.3836	0.0589
3	0.0021	0.0014	0.3993	0.0045
4	0.0051	0.5020	0.3970	0.0654
5				
6	-0.0069	-0.8329	0.4014	0.4510
7	-0.0102	-0.4493	0.3972	0.4865
8	-0.0003	0.0192	0.4050	0.4286
9	-0.0224	0.6013	0.4026	0.5067
10	0.0067	0.9000	0.3725	0.7039
11	-0.0058	-0.8466	0.3959	0.8406
12	0.0023	-0.5400	0.4005	0.8627
13	0.0134	-0.1357	0.4174	0.7441
14	-0.0037	0.5075	0.4025	0.8391
15	0.0113	0.8446	0.3910	0.9000

Table 4.6: Identified nonlinear coefficients for 10% error.

Case Number	E_1	E_3	η_1	η_3
1	-0.0068	-0.8856	0.3866	0.0376
2	0.0057	-0.5126	0.3775	0.0847
3	0.0019	0.0017	0.4004	0.0048
4	-0.0100	0.5543	0.3817	0.0626
5				
6	-0.0014	-0.8742	0.3980	0.4843
7	-0.0128	-0.4393	0.3976	0.4843
8	-0.0004	0.0228	0.4072	0.4118
9	-0.0384	0.6993	0.4116	0.4070
10	0.0186	0.7969	0.3769	0.6253
11	-0.0070	-0.8359	0.3959	0.8246
12	-0.0043	-0.4892	0.3977	0.8877
13	0.0010	-0.0402	0.4203	0.7199
14	-0.0028	0.4927	0.4025	0.8408
15	0.0150	0.8201	0.3923	0.9000

15. $E_3 = 0.9, \eta_3 = 0.9$

In order to better visualize the data, the difference was taken between the identified coefficient value and the real value. The data was arranged to determine any trends of the solver accuracy as the noise in the accelerometer increased for each identified coefficient.

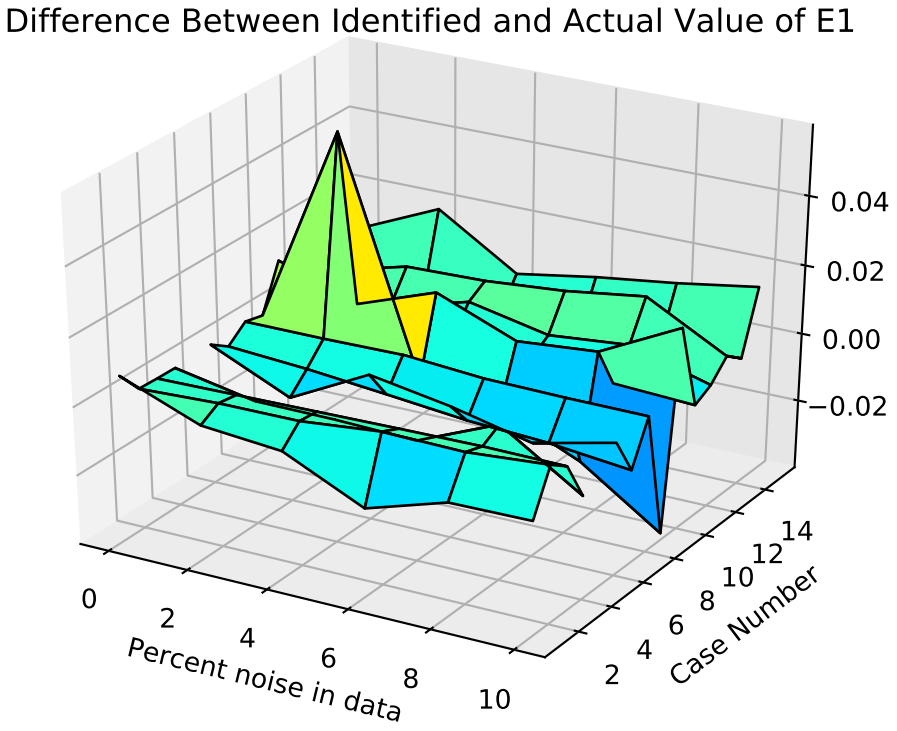

Figure 4.9: Surface plot of the difference between identified and actual values of E_1 for cases one through fifteen.

From the limits of the z-axis, it can be noted that the linear coating parameters, E_1 and η_1, were identified more accurately for all cases as compared to the nonlinear parameters as the difference between the identified and actual coefficient values are much smaller. It is possible that this is due to the fact that the minimization function was given an initial guess equal to the actual value for these parameters.

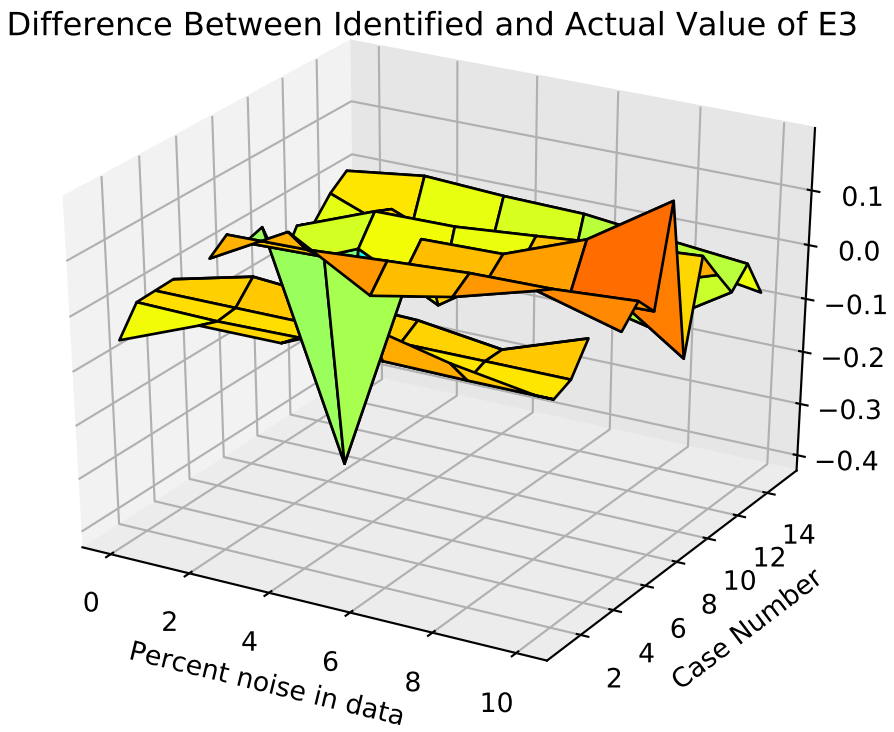

Figure 4.10: Surface plot of the difference between identified and actual values of E_3 for cases one through fifteen.

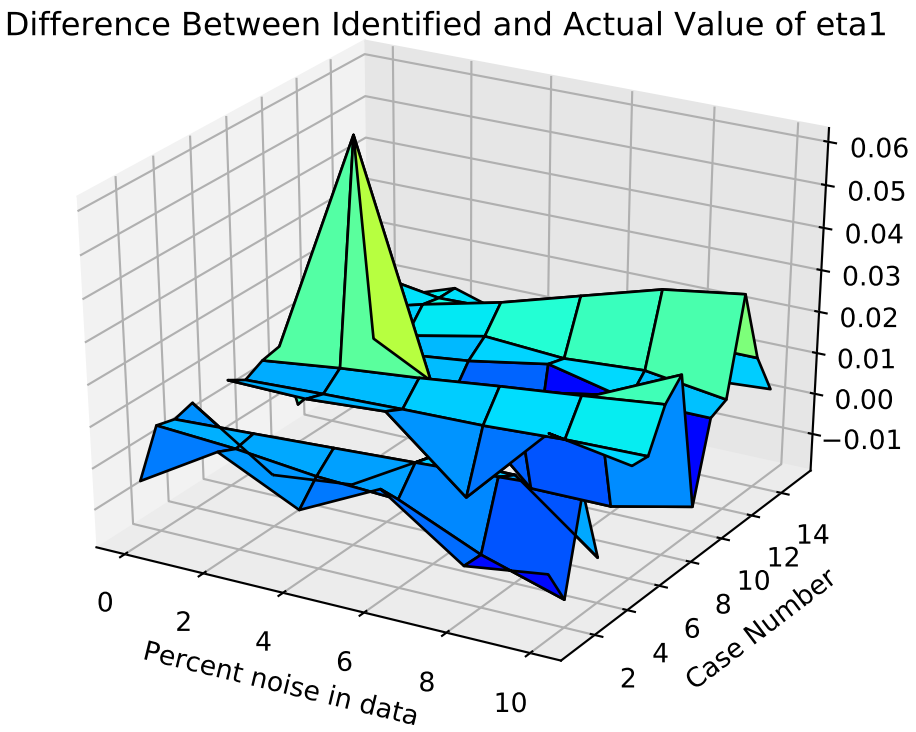

Figure 4.11: Surface plot of the difference between identified and actual values of η_1 for cases one through fifteen.

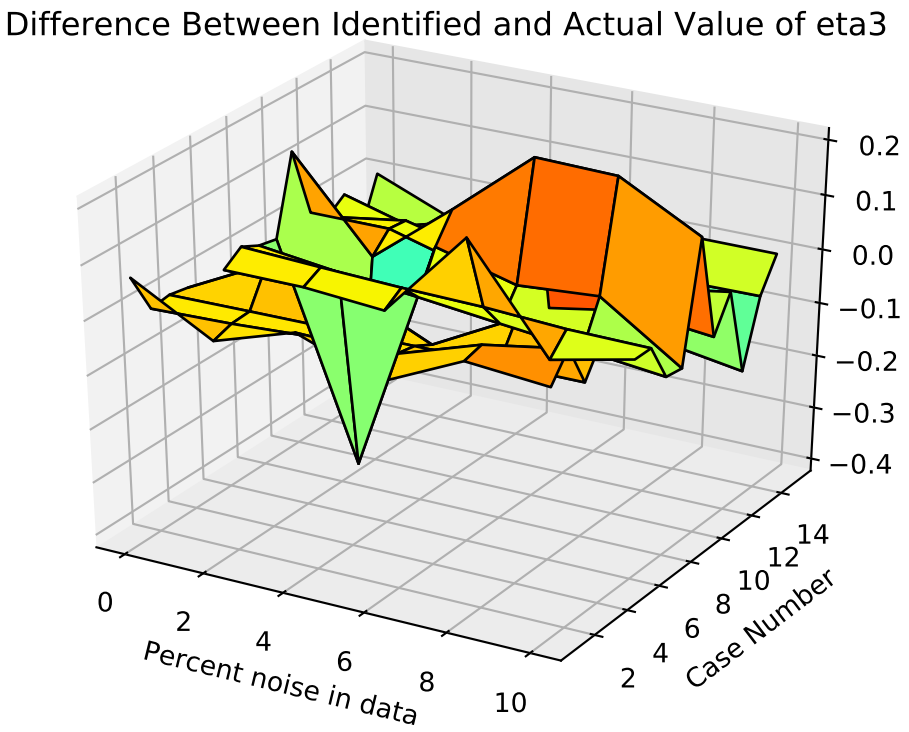

Figure 4.12: Surface plot of the difference between identified and actual values of η_3 for cases one through fifteen.

These figures also show that there is little change in solver accuracy based on the amount of noise in the accelerometer up until 10% noise. Anything greater than this level of noise would be unreasonable in testing. Therefore, noise in the data is not a significant factor in using the identification code.

However, there are still spikes from inaccurate solutions scattered randomly throughout the results including a gap where the solver failed completely. This indicates a general unreliability in the solver. The extent of this unreliability was examined by simulating approximately 75 cases for a single set of conditions as given below. The results are displayed as a set of histograms in Figures 4.13 through 4.16.

$E1 = 0$

$E3 = 0.5$

$eta1 = 0.39$

$eta3 = 0.5$

$noise = 5\%$

Again, it is apparent that the linear coating parameters were identified more accurately than the nonlinear parameters. The spread of the primary peak in the histogram is approximately an order of magnitude smaller for E_1 and η_1.

It is also notable that there are approximately five instances where the minimization function found an alternate local minimum by changing E_1 and η_1 rather than E_3 and η_3. It is recommended to inspect the results and subjectively assess whether or not an acceptable set of nonlinear coefficients has been identified. If not, the user can start the minimization over with a different initial guess based on the shape of the nonlinearity.

An alternate option would be to run the minimization function from a random set of initial guesses. With this approach, a set of local minimums will be found and from these the global minimum can be selected. This approach would prevent the selection of a local minimum as the identified coefficients. It would also likely address the gap where no solution was identified for case number five.

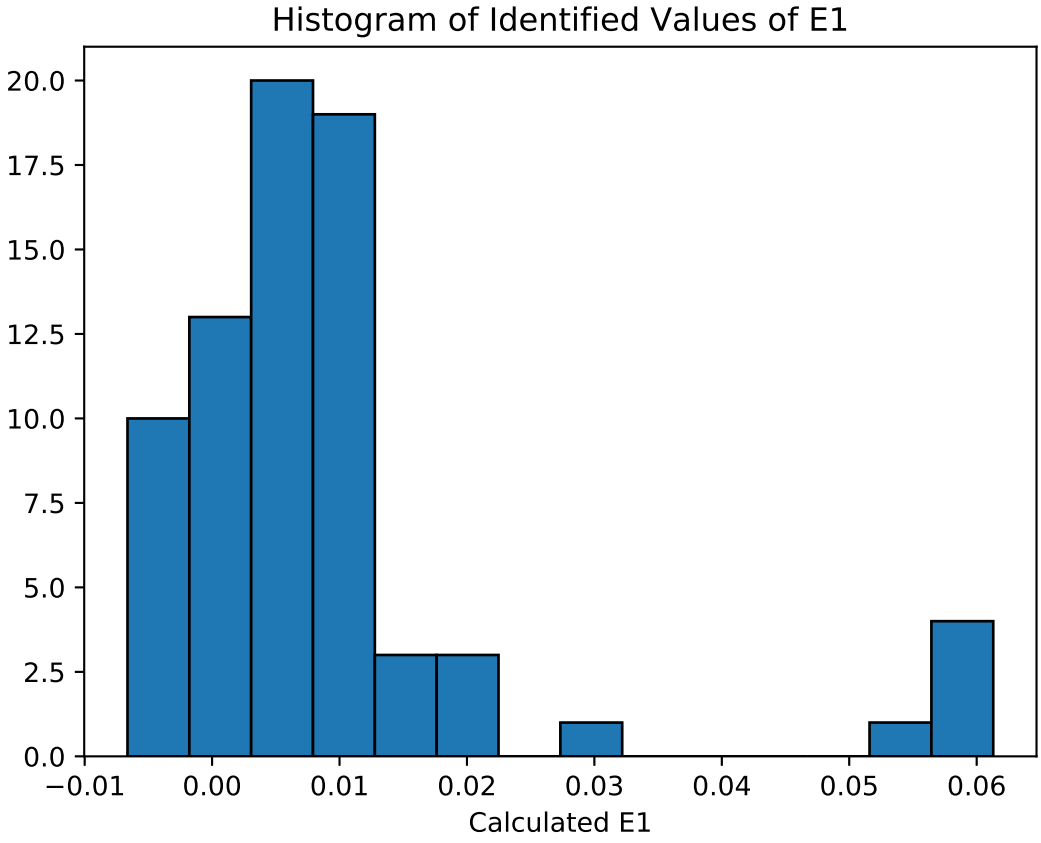

Figure 4.13: A histogram of identified values of E_1.

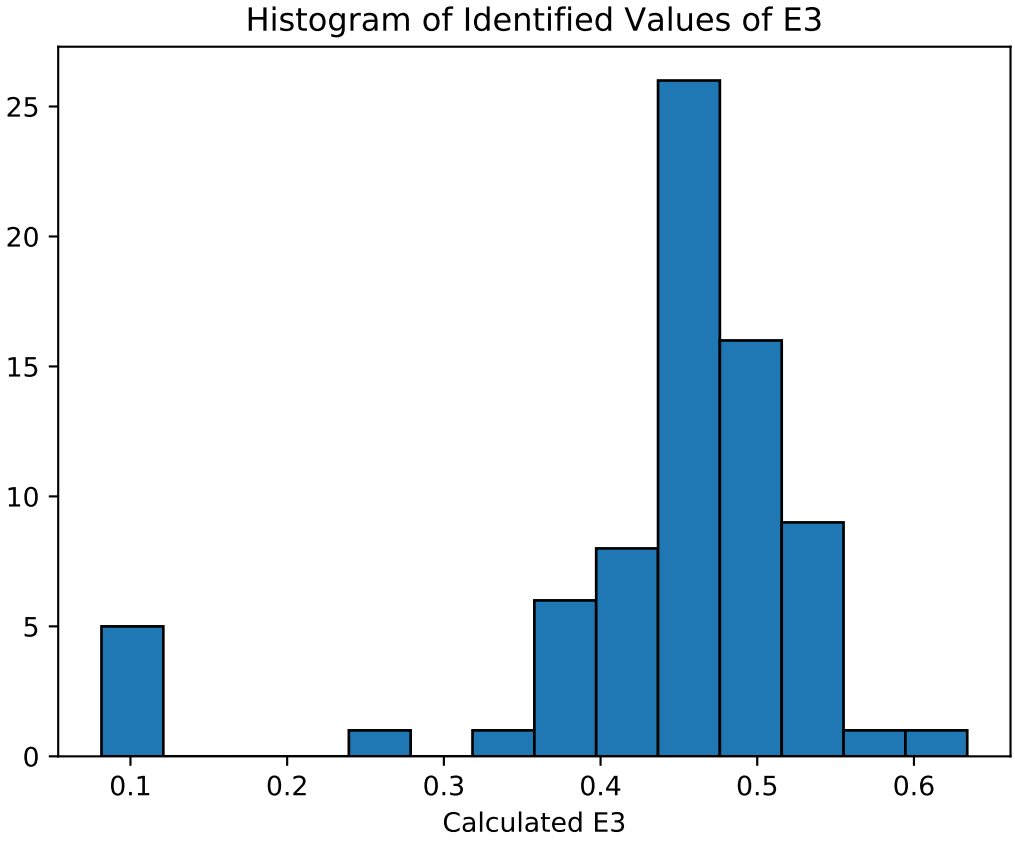

Figure 4.14: A histogram of identified values of E_3.

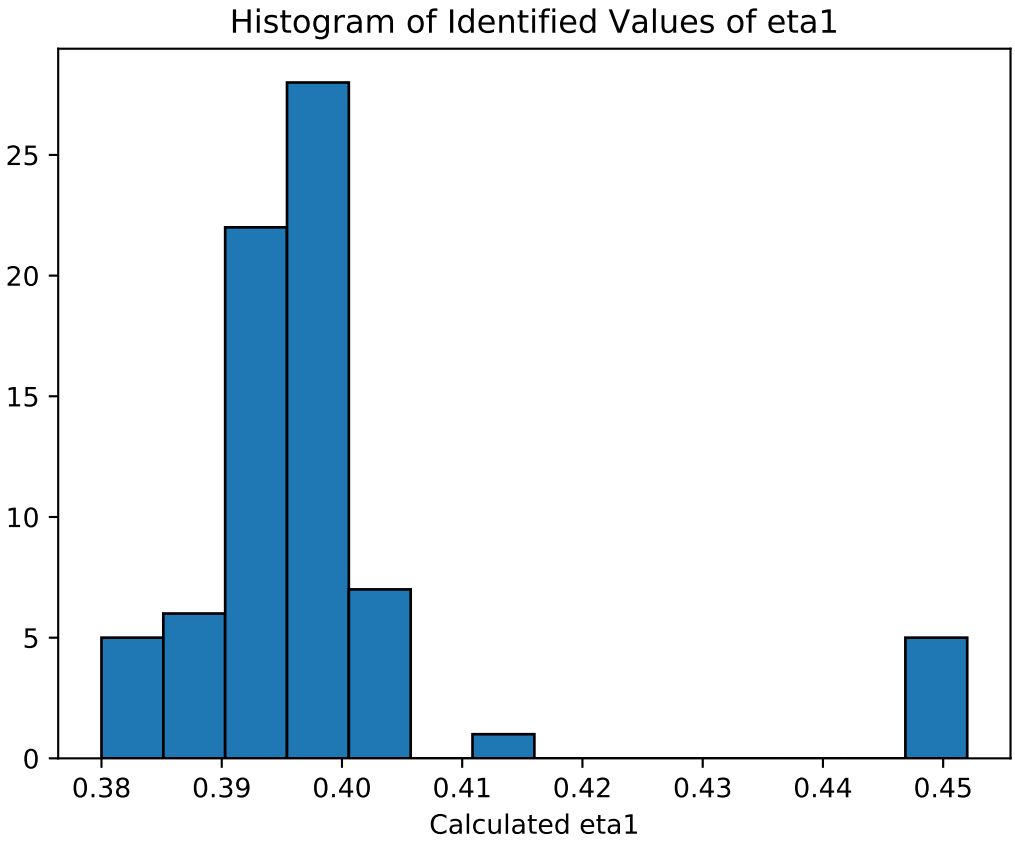

Figure 4.15: A histogram of identified values of η_1.

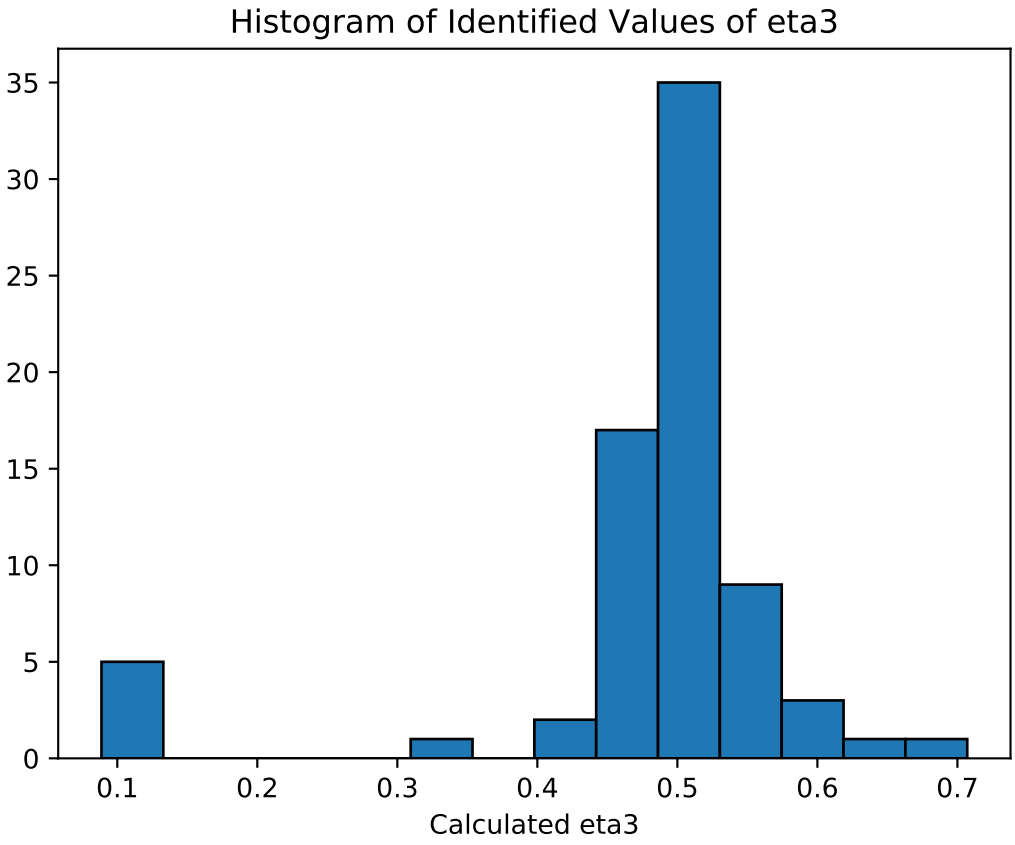

Figure 4.16: A histogram of identified values of η_3.

Applications

The method described in this work could eventually be applied to additionally determine variation in nonlinear coefficients with respect to temperature and coating thickness. With the nonlinear dissipative constitutive law coefficients that are identified, the full constitutive law can be incorporated into a nonlinear finite element model of the coating constructed in a Python based environment. A linear model of a blade can be constructed in a chosen commercial finite element code and then exported into a matrix based environment where the two models would be combined. The final, fully expanded model would be multi-physics based as it would incorporate nonlinearity due to amplitude of the cyclic strain, temperature, and thickness variation. The two models combined would form the systems EOM which can again be solved using the harmonic balance method described in Section 3.2.1.

Obtaining the forced response of a system is computationally expensive for nonlinear systems due to the iterative process. Unfortunately, this has negative ramifications for parametric studies and uncertainty analysis, both of which require a substantial increase in the number of solutions. A significantly computationally cheaper method is to obtain the solution of a system once for a parameter set and then to implement a continuation scheme.

Continuation schemes are based on the implicit function theorem which states that if the Jacobian matrix is non-singular, then for each parameter set in the vicinity of a fixed point a unique solution exists [15]. Continuation schemes take advantage of this theorem by using previously calculated solutions and assuming that the current solution is at a nearby

position. An optimization is then performed to minimize error. From this idea, the solution can be written as a function of the parameter set.

For the model, a continuation scheme can be used to determine the variation in the forced response as the amplitude of the cyclic strain changes. We expect that future efforts can use AUTO to implement an arc length tracing method to more rapidly determine the forced response for a range of parameters. [11]

Conclusions

Component life spans in gas turbines are notoriously difficult to predict. This is primarily due to the fact that the coatings applied to components to protect against corrosion and high temperatures also introduce nonlinear damping into the system. This research will be of interest to turbine manufactures for fatigue analysis improvements by enabling predictive modeling based on constitutive models.

This research used a power series expansion of Hooke's law in order to capture the effect of coating nonlinearity. The coefficients of the power series are determined from the EOM of a coated cantilever beam as compared to experimental results. The original intention was to obtain the experimental results from sine sweep curves through the second, third, and fourth bending mode. However, the experimental test setup used for this research had nonlinearity in the clamp which manifested in the results. After efforts to remove this nonlinearity proved unsuccessful, the experimental results were simulated. Future research efforts should focus on ensuring the experimental setup is free of extraneous nonlinear effects.

After the data was simulated, the nonlinear coefficients were identified for a test matrix of the E_3, η_3, and percent noise in the accelerometer. The coefficient identification code utilizes the harmonic balance method to solve the nonlinear EOMs. It was determined that reasonable amounts of noise in the data have minimal effects on the accuracy of the coefficient identification code. However, the identification code is prone to some reliability issues. In order to address this, it is proposed to run the solver for a range of random initial

guesses. This will address any instances where the solver finds a local minimum rather than the global minimum and even where the solver fails completely.

With the nonlinear coefficient identification procedure established, future research can focus on integrating the coating constitutive law into a finite element model of the turbine blade. This nonlinear model can again be solved using the harmonic balance method. Finally, design studies can be conducted by implementing a continuation method.

Bibliography

[1] Joseph C. Vibration testing, with modal analysis and health monitoring.

[2] F. Casadei, K. Bertoldi, and D.R. Clarke. Finite element study of multi-modal vibration damping for thermal barrier coating applications. *Computational Materials Science*, 79:908 – 917, 2013.

[3] M. J. Casiano. Extracting damping ratio from dynamic data and numerical solutions. Technical report, National Aeronautics and Space Administration, 2016.

[4] David R. Clarke, Matthias Oechsner, and Nitin P. Padture. Thermal-barrier coatings for more efficient gas-turbine engines. *MRS Bulletin*, 37(10):891–898, 2012.

[5] K. R. Cross, W. R. Lull, R. L. Newman, and andJ. R. Cavanagh. Potential of graded coatings in vibration damping. *Journal of Aircraft*, 10(11):689–691, November 1973.

[6] Sergio Filippi and Peter J. Torvik. A methodology for predicting the response of blades with non-linear coatings. In *Proceedings of the ASME Turbo Expo*, volume 6, pages 729 – 737, Glasgow, United kingdom, 2010.

[7] Sergio Filippi and Peter J. Torvik. A methodology for predicting the response of blades with nonlinear coatings. *Journal of Engineering for Gas Turbines and Power*, 133(4), 2011.

[8] Nancy Hall. Gas turbine propulsion. NASA Glenn Research Center, May 2015.

[9] Daniel J. Inman and Ramesh C. Singh. *Engineering Vibration*. Prentice Hall, 2001.

[10] Frank Ivancic, Anthony Palazotto, and Charles Cross. The effect of a hard coating on the damping and fatigue life of titanium. In *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, volume 4, pages 2604 – 2613, Norfolk, VA, United states, 2003.

[11] Pankaj Kamthan. Auto software for continuation and bifurcation problems in ordinary differential equations, 1995.

[12] Andi M. Limarga, Tuan L. Duong, Giuliano Gregori, and David R. Clarke. High-temperature vibration damping of thermal barrier coating materials. *Surface and Coatings Technology*, 202(4-7):693 – 697, 2007.

[13] S. Madhavan, Rajeev Jain, C. Sujatha, and A.S. Sekhar. Vibration based damage detection of rotor blades in a gas turbine engine. *Engineering Failure Analysis*, 46:26 – 39, 2014.

[14] W. Maktouf, K. Ammar, I. Ben Naceur, and K. Sai. Multiaxial high-cycle fatigue criteria and life prediction: Application to gas turbine blade. *International Journal of Fatigue*, 92:25 – 35, 2016.

[15] Ali H. Nayfeh and Balakumar Balachandran. *Applied Nonlinear Dynamics*. WILEY-VCH Verlag GmbH & Co. KGaA, 2004.

[16] R. Rajendran. Gas turbine coatings - an overview. *Engineering Failure Analysis*, 26:355 – 369, 2012.

[17] S.H. Raza, M.A. Malik, and W. Akram. Analysis of material coating for damping in beam structures. In *Key Engineering Materials*, volume 442, pages 202 – 210, Islamabad, Pakistan, 2010.

[18] Joseph Slater. mousai, October 2017. Online. Accessed 26 November 2017.

[19] N. Tassini, K. Lambrinou, I. Mircea, M. Bartsch, S. Patsias, and O. Van der Biest. Study of the amplitude-dependent mechanical behaviour of yttria-stabilised zirconia thermal barrier coatings. *Journal of the European Ceramic Society*, 27(2-3):1487 – 1491, 2007.

[20] P. Torvik, R. Willson, and J. Hansel. Influence of a viscoelastic surface infiltrate on the damping properties of plasma sprayed alumina coatings part i: room temperature. In *Materials Science and Technology Conference and Exhibition, MS and T'07 - Exploring Structure, Processing, and Applications Across Multiple Materials Systems*, volume 3, pages 2051 – 2062, Detroit, MI, United states, 2007.

[21] P. Torvik, R. Willson, J. Hansel, and J. Henderson. Influence of a viscoelastic surface infiltrate on the damping properties of plasma sprayed alumina coatings part ii: effects of elevated temperatures and static strain. In *Materials Science and Technology Conference and Exhibition, MS and T'07 - Exploring Structure, Processing, and Applications Across Multiple Materials Systems*, volume 3, pages 2063 – 2074, Detroit, MI, United states, 2007.

[22] Peter J. Torvik. Analysis of free-layer damping coatings. In *Key Engineering Materials*, volume 333, pages 195 – 214, 2007.

[23] Peter J. Torvik. A survey of the damping properties of hard coatings for turbine engine blades. In *Integration of Machinery Failure Prevention Technologies into Systems Health Management - Proceedings of the 61st Meeting of the Society for Machinery Failure Prevention Technology*, pages Office of Naval Research; U.S. Army Research Laboratory; National Aeronautics and Space Administration; Naval Surface Warfare Center; Air Force Research Laboratory (AFRL) –, Virginia Beach, VA, United states, 2007.

[24] Peter J. Torvik. Coatings for enhanced passive damping. In *Ceramic Engineering and Science Proceedings*, volume 29, pages 3 – 18, Daytona Beach, FL, United states, 2009.

[25] Peter J. Torvik. Determination of mechanical properties of non-linear coatings from measurements with coated beams. *International Journal of Solids and Structures*, 46(5):1066 – 1077, 2009.

[26] Peter J. Torvik and Jason Hansel. Mechanical properties of a ceramic coating with vem infiltration. *Journal of Engineering Materials and Technology, Transactions of the ASME*, 131(3):0310031 – 0310039, 2009.

[27] Peter J. Torvik and Bryan Langley. Material properties of hard coatings developed for high damping. In *51st AIAA/SAE/ASEE Joint Propulsion Conference*, Orlando, FL, United states, 2015.

[28] P.J. Torvik. A note on the estimation of nonlinear system damping. *Journal of Applied Mechanics, Transactions ASME*, 70(3):449 – 450, 2003.

[29] P.J. Torvik and J.P. Henderson. Influence of glass content on damping properties of plasma-sprayed mixtures of zirconia and glass. *Journal of Materials Engineering and Performance*, 21(7):1405 – 1415, 2012.

Appendix A: Sine Sweep Simulation Code

Included below is the python code used to simulate the experimental data for this research effort.

```
1
2  import numpy as np
3  import pylab as pl
4  import scipy as sp
5  import os
6  from scipy.integrate import odeint
7  from scipy.fftpack import fft
8  from UtilityFunctions import PositionCoeff, PlotFRF,
       SamplingTimes, SystemWrapper
9
10 global l, b, h, Cthickness, I, A, Ac, Eb, rho, in2m
11 l = 6 #inches
12 b = 0.752 #inches
13 h = 0.0632 #inches
14 Cthickness = 0.0034 #inches
15 rho = 4231 #kg/m^3
16 Eb = 10.42e10 * 1.00012571
17 c = 0.65 * 0.8061281 #linear damping coefficient
```

```
18 in2m = 0.0254 #Conversion factor (inches to meters)
19 l = in2m*l
20 b = in2m*b
21 h = in2m*h
22 Cthickness = in2m*Cthickness
23 I = 1/12*b*h**3
24 A = b*h #Cross sectional area
25 Ac = b*Cthickness #Coating cross sectional area
26
27 def SineSweepSimulation(NonlinearCoeffs,folder):
28 #    NonlinearCoeffs=sp.array([E1,E2,E3,eta1,eta2,eta3,alpha1
       ])
29       os.mkdir(folder)
30       CtrlAccel = sp.array([0.1,0.25,0.5])
31       df = 0.2
32       Freq = np.arange(1160,1180,df)
33       e_num = 5
34       W = np.zeros([e_num,np.size(CtrlAccel),np.size(Freq)],
           dtype=complex)
35       X1, X2, X3 = PositionCoeff()
36       E1 = NonlinearCoeffs[0]
37       E2 = NonlinearCoeffs[1]*0
38       E3 = NonlinearCoeffs[2]
39       eta1 = NonlinearCoeffs[3]
40       eta2 = NonlinearCoeffs[4]*0
41       eta3 = NonlinearCoeffs[5]
42       alpha1 = NonlinearCoeffs[6]*0
```

```
43      Tguess=[0,0,0,0,0,0]
44      dwell_time = 2
45      n_ppc = 25
46      FreqR = 2*np.pi*Freq
47      Base_Displacement = np.outer(CtrlAccel,-9.81/FreqR**2)
48      X1, X2, X3 = PositionCoeff()
49      for j in range(len(CtrlAccel)):
50          Base_Displacement_j = Base_Displacement[j,:]
51          for i in range(len(Freq)):
52              w = FreqR[i]
53              BCoeff = Base_Displacement_j[i]
54              t = SamplingTimes(tend_approx = dwell_time, Freq
                    = Freq[i], n_ppc = n_ppc)
55              sol = odeint(SystemWrapper, Tguess, t, args = (w
                    ,BCoeff,E1,E2,E3,eta1,eta2,eta3,alpha1))
56              sol2 = np.transpose(sol[-(n_ppc+1):-1,:])
57              sol3 = X1 * sol2[0,:] + X2 * sol2[1,:] + X3 *
                    sol2[2,:]
58              err = (2*np.random.rand(1,len(sol3))-1) * 0.005e
                    -3 #max(sol3)
59              fftlen = len(t[-(n_ppc+1):-1])
60              for k in range(e_num):
61                  xhar = fft(sol3 + err*k*0.2) * 2 / fftlen
62  #                   W[k,j,i] = xhar[1]
63                  W[k,j,i] = xhar[0,1]
64              Tguess = sol[-1,:]
65      for l in range(e_num):
```

```
66          pl.close('all')
67          PlotFRF(Freq = Freq, W = W[l,:,:])
68          pl.savefig(folder+'/FRF_'+str(l*2)+'.png')
69          np.savetxt(folder+'/FRF_Surface_Real_'+str(l*2)+'.
                txt',np.real(W[l,:,:]))
70          np.savetxt(folder+'/FRF_Surface_Imag_'+str(l*2)+'.
                txt',np.imag(W[l,:,:]))
71      return
```

Appendix B: Theoretical FRF Generation and Minimization Implementation

Included below is the python python code used to identify the nonlinear coating coefficients for this research effort.

```
1
2  import numpy as np
3  import pylab as pl
4  import scipy.io as sio
5  import scipy.optimize as sci
6  import scipy as sp
7  import time
8  from UtilityFunctions import ImportData, PositionCoeff,
       ReplaceNANs, ReplaceNANsReguess, StateSpaceForm
9  #import mousai as ms
10 from har_bal import hb_freq
11
12 '''
13
14 '''
15
```

```
16 global l, b, h, Cthickness, I, A, Ac, Eb, rho, in2m
17 l = 6 #Beam length in inches
18 b = 0.752 #Beam width in inches
19 h = 0.0632 #Beam height in inches
20 Cthickness = 0.0034 #Coating thickness in inches
21 rho = 4231 #Beam density in kg/m^3
22 Eb = 10.42e10 * 1.00012571 #Beam Young's modulus
23 c = 0.65 * 0.8061281 #Coating linear damping coefficient
24
25 in2m = 0.0254 #Conversion factor (inches to meters)
26 #Convert all dimensions from inches to meters
27 l = in2m*l
28 b = in2m*b
29 h = in2m*h
30 Cthickness = in2m*Cthickness
31
32 I = 1/12*b*h**3 #Beam moment of inertia
33 A = b*h #Beam cross sectional area
34 Ac = b*Cthickness #Coating cross sectional area
35
36
37 def FindNonlinearFRF_DataRange(NonlinearCoeffs,n,Freq,FreqR,
       CtrlAccel, num_steps = 7):
38     '''
39
40     Parameters
41     ----------
```

```
42
43      Returns
44      -------
45
46      Examples
47      --------
48
49      Notes
50      -----
51
52      '''
53      F_size = np.size(Freq)
54      C_size = np.size(CtrlAccel)
55      Tguess = np.zeros([6,2*n + 1])
56      T_freq = np.ones([C_size,6,F_size])
57      Phase_freq = np.ones([C_size,6,F_size])
58      E1 = NonlinearCoeffs[0]
59      E2 = 0
60      E3 = NonlinearCoeffs[1]
61      eta1 = NonlinearCoeffs[2]
62      eta2 = 0
63      eta3 = NonlinearCoeffs[3]
64      alpha1 = 0
65      Base_Displacement = np.outer( CtrlAccel, 1 / FreqR**2 )
            * -9.81
66      for j, baseint in enumerate(Base_Displacement, start=0):
67          Base_Displacement_j = Base_Displacement[j,:]
```

```
68
69          for i, fint in enumerate(Freq, start=0):
70              w = FreqR[i]
71              BCoeff = Base_Displacement_j[i]
72              params = {'omega':w,'BCoeff':BCoeff,'E1':E1,'E2'
                    :E2,'E3':E3,'eta1':eta1,'eta2':eta2,'eta3':
                    eta3,'alpha1':alpha1}
73  #              _,Tguess,_,amps,phases = ms.har_bal.hb_freq(
    StateSpaceForm,x0=Tguess,omega=w,num_harmonics=n,eqform='
    first_order',params=params,num_time_steps=num_steps)
74              _,Tguess,_,amps,phases = hb_freq(StateSpaceForm,
                    x0=Tguess,omega=w,num_harmonics=n,eqform='
                    first_order',params=params,num_time_steps=
                    num_steps)
75              T_freq[j,:,i] = amps
76              Phase_freq[j,:,i] = phases
77      T_complex = T_freq * np.exp(1j * Phase_freq)
78      _,T_complex,_ = ReplaceNANsReguess(T_complex,Freq,FreqR,
            Base_Displacement,params,CtrlAccel,n,num_steps)
79
80      X1, X2, X3 = PositionCoeff()
81      W = X1 * T_complex[:,0,:] + X2 * T_complex[:,1,:] + X3 *
            T_complex[:,2,:]
82      W = sp.reshape(W,[C_size,F_size])
83      return W
84
```

```
85 def CompareExperimentTheoretical(NonlinearCoeffs,n,Freq,
       FreqR,CtrlAccel,W_Simulated):
86     W = FindNonlinearFRF_DataRange(NonlinearCoeffs=
           NonlinearCoeffs, n=n, Freq=Freq, FreqR=FreqR,
           CtrlAccel=CtrlAccel)
87     _,W,_ = ReplaceNANs(W)
88     R = abs(W - W_Simulated)*1e6
89     return sum(sum(R**2))
90
91 def FitToNonlinearCurve(name='E1_0_E3_0_eta1_39_eta3_10\
       FRF_Surface',n=1,noise = 0):
92     '''
93
94     Parameters
95     ----------
96
97     Returns
98     -------
99
100    Examples
101    --------
102
103    Notes
104    -----
105
106    '''
```

```
107         Freq, FreqR, CtrlAccel, W_Simulated = ImportData(name =
                name, noise = noise)
108         time.sleep(5*60)
109 #       NonlinearCoeffs = sp.array([E1, E3, eta1, eta3])
110         NonlinearCoeffs = sp.array([0,0,0.65/1.65,0])
111         bnds = ((-1,0.9),(-3.4/4.9/0.7,0.9),(0,0.9),(-0.2,0.9))
112
113         #SETUP AND MINIMIZATION FUNCTION
114         optimizedResults = sci.minimize(
                CompareExperimentTheoretical, NonlinearCoeffs, \
115         args=(n,Freq,FreqR,CtrlAccel,W_Simulated), method='L-
                BFGS-B', bounds=bnds, \
116         options={'disp': True,'ftol':1e-3,'eps':1e-4, 'maxls'
                :10})
117         NonlinearCoeffs_calc = optimizedResults.x
118         return NonlinearCoeffs_calc
```

Appendix C: Utility Functions Used Throughout

Included below is the python code for several functions used in both Appendix A and B and generally throughout the research effort.

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Wed Aug  8 20:20:25 2018
4
5  @author: kirby
6  """
7
8  import numpy as np
9  import pylab as pl
10 import scipy as sp
11 #import mousai as ms
12 from har_bal import hb_freq
13
14 global l, b, h, Cthickness, I, A, Ac, Eb, rho, in2m
15 l = 6 #Beam length in inches
16 b = 0.752 #Beam width in inches
17 h = 0.0632 #Beam height in inches
18 Cthickness = 0.0034 #Coating thickness in inches
```

```
19 rho = 4231 #Beam density in kg/m^3
20 Eb = 10.42e10 * 1.00012571
21 c = 0.65 * 0.8061281 #linear damping coefficient
22 in2m = 0.0254 #Conversion factor (inches to meters)
23 l = in2m*l
24 b = in2m*b
25 h = in2m*h
26 Cthickness = in2m*Cthickness
27 I = 1/12*b*h**3
28 A = b*h #Cross sectional area
29 Ac = b*Cthickness #Coating cross sectional area
30
31 #
    ##########################################################################

32 def PlotFRF(Freq, W):
33     pl.figure()
34     pl.subplot(2,1,1)
35     for i in range(len(W)): pl.plot(Freq,1000*abs(W[i,:]),'.
           ')
36     pl.xlabel('Frequency (Hz)')
37     pl.ylabel('Amplitude(mm)')
38     pl.subplot(2,1,2)
39     for i in range(len(W)): pl.plot(Freq, 180/np.pi*np.angle
           (W[i,:]),'.')
40     pl.xlabel('Frequency (Hz)')
41     pl.ylabel('Phase (deg)')
```

```
42      return
43
44  def PositionCoeff(x = 2.75):
45      x = x * in2m

            #Position of accelerometer
46      B = np.array([1.87510407/1, 4.69409113/1, 7.85475744/1,
            10.99554073/1])
47      SigmaN = np.array([0.7341, 1.0185, 0.9992, 1, 1])
48      X1 = np.cosh(B[1] * x) - np.cos(B[1] * x) - SigmaN[1] *
            (np.sinh(B[1] * x) - np.sin(B[1] * x))
49      X2 = np.cosh(B[2] * x) - np.cos(B[2] * x) - SigmaN[2] *
            (np.sinh(B[2] * x) - np.sin(B[2] * x))
50      X3 = np.cosh(B[3] * x) - np.cos(B[3] * x) - SigmaN[3] *
            (np.sinh(B[3] * x) - np.sin(B[3] * x))
51      return X1, X2, X3
52
53  def StateSpaceForm(T,params):
54      dT = np.array([[T[3]],[T[4]],[T[5]]])
55      T = np.array([[T[0]],[T[1]],[T[2]]])
56      R = FirstThreeBendingModesEOM(T,dT,params)
57      return np.array([dT[0],dT[1],dT[2],R[0],R[1],R[2]])
58
59  #
        ##########################################################################

60  def Combine(T_freq, Phase_freq):
```

```
61      X1, X2, X3 = PositionCoeff()
62      T_complex = T_freq * np.exp(1j * Phase_freq)
63      W = X1 * T_complex[:,0,:] + X2 * T_complex[:,1,:] + X3 *
            T_complex[:,2,:]
64      return W
65
66  def SamplingTimes(tend_approx, Freq, n_ppc = 25):
67      T = 1/Freq
68      n_periods = sp.floor(tend_approx/T)
69      tend = n_periods*T
70      num = n_ppc*n_periods + 1
71      t = np.linspace(0, tend, int(num))
72      return t
73
74  def SystemWrapper(T,t,w,BCoeff,E1,E2,E3,eta1,eta2,eta3,
        alpha1):
75      params = {'omega':w,'BCoeff':BCoeff,'E1':E1,'E2':E2,'E3'
            :E3,'eta1':eta1,'eta2':eta2,'eta3':eta3,'alpha1':
            alpha1}
76      params['cur_time'] = t
77      R = StateSpaceForm(T,params)
78      return R.reshape(6)
79
80  #
        ##################################################################
```

```
81 def ImportData(name='E3_4e17_eta3_4e5\FRF_Surface',noise =
       0):
82 #     W_Simulated = np.loadtxt(name)
83       W_Simulated_Real = np.genfromtxt(name + '_Real_' + str(
             noise) + '.txt')
84       W_Simulated_Imag = np.genfromtxt(name + '_Imag_' + str(
             noise) + '.txt')
85       W_Simulated = W_Simulated_Real + 1j * W_Simulated_Imag
86       df = 0.2
87       Freq = np.arange(1160,1180,df)
88       FreqR = 2 * np.pi * Freq
89       CtrlAccel = sp.array([0.1,0.25,0.5])
90       return Freq, FreqR, CtrlAccel, W_Simulated#Displacement,
             phi
91
92 #Freq,FreqR,CtrlAccel, W_Simulated = ImportData(name='
       ReliabilityData\E1_0_E3_50_eta1_39_eta3_5018\FRF_Surface
       ',noise = 5)
93 #PlotFRF(Freq,W_Simulated)
94
95 def ReplaceNANs(x):
96       x_old = x
97       #Assumes no nan's in the first/last elements in each
             matrix
98       inc = 0
99       for i in range(x.shape[0]):
100          for j in range(x.shape[1]):
```

```
101                 if np.isnan(x[i,j]) == True:
102                     #FREQUENCY ACROSS ROWS
103                     n = j+1
104                     while np.isnan(x[i,n]) == True: n = n + 1
105                     diff_mag = abs(x[i,n]) - abs(x[i,j-1])
106                     diff_phase = np.angle(x[i,n]) - np.angle(x[i
                            ,j-1])
107                     for count, k in enumerate(np.linspace(j,n-1,
                            num=(n-j)),start=1):
108                         int_mag = count/(n-j+1) * diff_mag + abs
                                (x[i,j-1])
109                         int_phase = count/(n-j+1) * diff_phase +
                                 np.angle(x[i,j-1])
110                         x[i,int(k)] = int_mag * np.exp( 1j *
                                int_phase )
111                         inc = inc + 1
112         if inc > 0: print('Interpolated ' + str(inc) + ' NAN
                values')
113         return x_old, x, inc
114
115 def ReplaceNANsReguess(x,Freq,FreqR,Base_Displacement,params
        ,CtrlAccel,num_har=1,num_steps=5):
116         x_old = x
117         #Assumes no nan's in the first/last elements in each
                matrix
118         inc = 0
119         for i in range(x.shape[0]):
```

```
120         for j in range(x.shape[2]):
121             x_isnan = np.isnan(x[i,:,:])
122             if x_isnan[:,j].any() == True and x_isnan[:,j
                    -1].any() == False:
123                 #FREQUENCY ACROSS ROWS
124                 n = j+1
125                 while x_isnan[:,n].any() == True: n = n + 1
126                 for count, k in enumerate(np.linspace(n-1,j,
                        num=(n-j),dtype=np.int8),start=1):
127                     k_real = k
128                     while np.isnan(x[i,0,k_real+1]) == True:
                            k_real = k_real+1
129                     diff_mag = abs(x[i,:,k_real+1]) - abs(x[
                        i,:,j-1])
130                     diff_phase = np.angle(x[i,:,k_real+1]) -
                            np.angle(x[i,:,j-1])
131                     int_mag = (k-j+1)/(k_real-j+2) *
                        diff_mag + abs(x[i,:,j-1])
132                     int_phase = (k-j+1)/(k_real-j+2) *
                        diff_phase + np.angle(x[i,:,j-1])
133
134                     w = FreqR[k]
135                     params['omega'] = w
136                     BCoeff = Base_Displacement[i,k]
137                     params['BCoeff'] = BCoeff
138                     time = np.linspace(0, 2 * np.pi / w, num
                        =1+2*num_har, endpoint=False)
```

```
139                         Tguess = np.real(np.outer(int_mag*np.exp
                                (1j*(int_phase)),np.exp(1j*(w*time)))
                                )
140 #                        _,_,_,amps_new,phases_new = ms.har_bal.
    hb_freq(StateSpaceForm,x0=Tguess,omega=w,num_harmonics=
    num_har,eqform='first_order',params=params,num_time_steps
    =num_steps)
141                         _,_,_,amps_new,phases_new = hb_freq(
                                StateSpaceForm,x0=Tguess,omega=w,
                                num_harmonics=num_har,eqform='
                                first_order',params=params,
                                num_time_steps=num_steps)
142
143                         x[i,:,k] = amps_new * np.exp(1j *
                                phases_new)
144                         inc = inc + 1
145     if inc > 0: print('Recalculated ' + str(inc) + ' NAN
            values')
146     return x_old, x, inc
147
148 def FirstThreeBendingModesEOM(T,dT,params):
149     #Displacement and velocity values of first three modes
150     T1 = T[0]
151     T2 = T[1]
152     T3 = T[2]
153     dT1 = dT[0]
154     dT2 = dT[1]
```

```
155     dT3 = dT[2]
156     #Nonlinear coefficients
157     E1 = params['E1'] * 1e9 + 1.4275e11
158     E2 = params['E2']*0
159     E3 = params['E3'] * 4.9e17 * 0.7
160     eta1 = params['eta1'] * 1.65e5
161     eta2 = params['eta2']*0
162     eta3 = params['eta3'] * 1e6
163     alpha1 = params['alpha1']*1e15
164
165     #Additional miscellaneous parameters
166     omega = params['omega'] #Excitation frequency
167     BCoeff = params['BCoeff'] #Displacement amplitude of the
            base
168     cur_time = params['cur_time'] #Current time
169     F0 = sp.array([BCoeff*np.imag(np.exp(1j*(omega*cur_time
          - np.pi/2)))]) #Displacement of base
170
171     M = np.matrix([[      -1.0003807372243,
          0.00367802363230054,     0.10642562620933], \
172                    [ 0.00367802362921381,     -1.0040833046823,
                         -0.0709318110613687], \
173                    [     0.10642562553913, -0.0709318078588694,
                         -0.954530027467067]]) * (A * l * rho)
174     R = np.zeros([3,1])
175     #Acceleration of first three modes
176     R[0,:] = 485.703672369963*Eb*I*T1/l**3 \
```

```
177                 - 14.0005671307443*Eb*I*T2/l**3 \
178                 - 1555.65246221105*Eb*I*T3/l**3 \
179                 + 1.00038073722512*c*l*dT1 \
180                 - 0.00367802364962702*c*l*dT2 \
181                 - 0.106425625320298*c*l*dT3 \
182                 - 0.433569433920057*A*F0*l*omega**2*rho
183         R[1,:] = -0.892874841326496*Ac*E1*h**2*T1/l**3 \
184                 + 1911.04477891575*Ac*E1*h**2*T2/l**3 \
185                 + 518.414833236068*Ac*E1*h**2*T3/l**3 \
186                 + 4252.52966664376*Ac*E2*h**3*T1**2/l**5 \
187                 + 32250.972671513*Ac*E2*h**3*T2**2/l**5 \
188                 + 185695.881715308*Ac*E2*h**3*T3**2/l**5 \
189                 + 22207.3904995383*Ac*E3*h**4*T1**3/l**7 \
190                 + 3070774.06162269*Ac*E3*h**4*T2**3/l**7 \
191                 + 640811451.309187*Ac*E3*h**4*T3**3/l**7 \
192                 + 22207.3904995383*Ac*alpha1*h**4*T1**2*dT1/l**7
                        \
193                 + 3070774.06162269*Ac*alpha1*h**4*T2**2*dT2/l**7
                        \
194                 + 640811451.309187*Ac*alpha1*h**4*T3**2*dT3/l**7
                        \
195                 - 0.892874841326496*Ac*eta1*h**2*dT1/l**3 \
196                 + 1911.04477891575*Ac*eta1*h**2*dT2/l**3 \
197                 + 518.414833236068*Ac*eta1*h**2*dT3/l**3 \
198                 + 4252.52966664376*Ac*eta2*h**3*dT1**2/l**5 \
199                 + 32250.972671513*Ac*eta2*h**3*dT2**2/l**5 \
200                 + 185695.881715308*Ac*eta2*h**3*dT3**2/l**5 \
```

```
201             + 22207.3904995383*Ac*eta3*h**4*dT1**3/l**7 \
202             + 3070774.06162269*Ac*eta3*h**4*dT2**3/l**7 \
203             + 640811451.309187*Ac*eta3*h**4*dT3**3/l**7 \
204             - 1.78574968265299*Eb*I*T1/l**3 \
205             + 3822.0895578315*Eb*I*T2/l**3 \
206             + 1036.82966647214*Eb*I*T3/l**3 \
207             - 0.00367802364962702*c*l*dT1 \
208             + 1.00408330467803*c*l*dT2 \
209             + 0.0709318096643847*c*l*dT3 \
210             - 0.258438996357548*A*F0*l*omega**2*rho
211     R[2,:] = -51.6716437190771*Eb*I*T1/l**3 \
212             + 270.005185993919*Eb*I*T2/l**3 \
213             + 13952.626911255*Eb*I*T3/l**3 \
214             - 0.106425625320298*c*l*dT1 \
215             + 0.0709318096643847*c*l*dT2 \
216             + 0.954530027232067*c*l*dT3 \
217             - 0.272839404111756*A*F0*l*omega**2*rho
218     R = np.linalg.inv(M) @ R
219     return np.array(R)
```